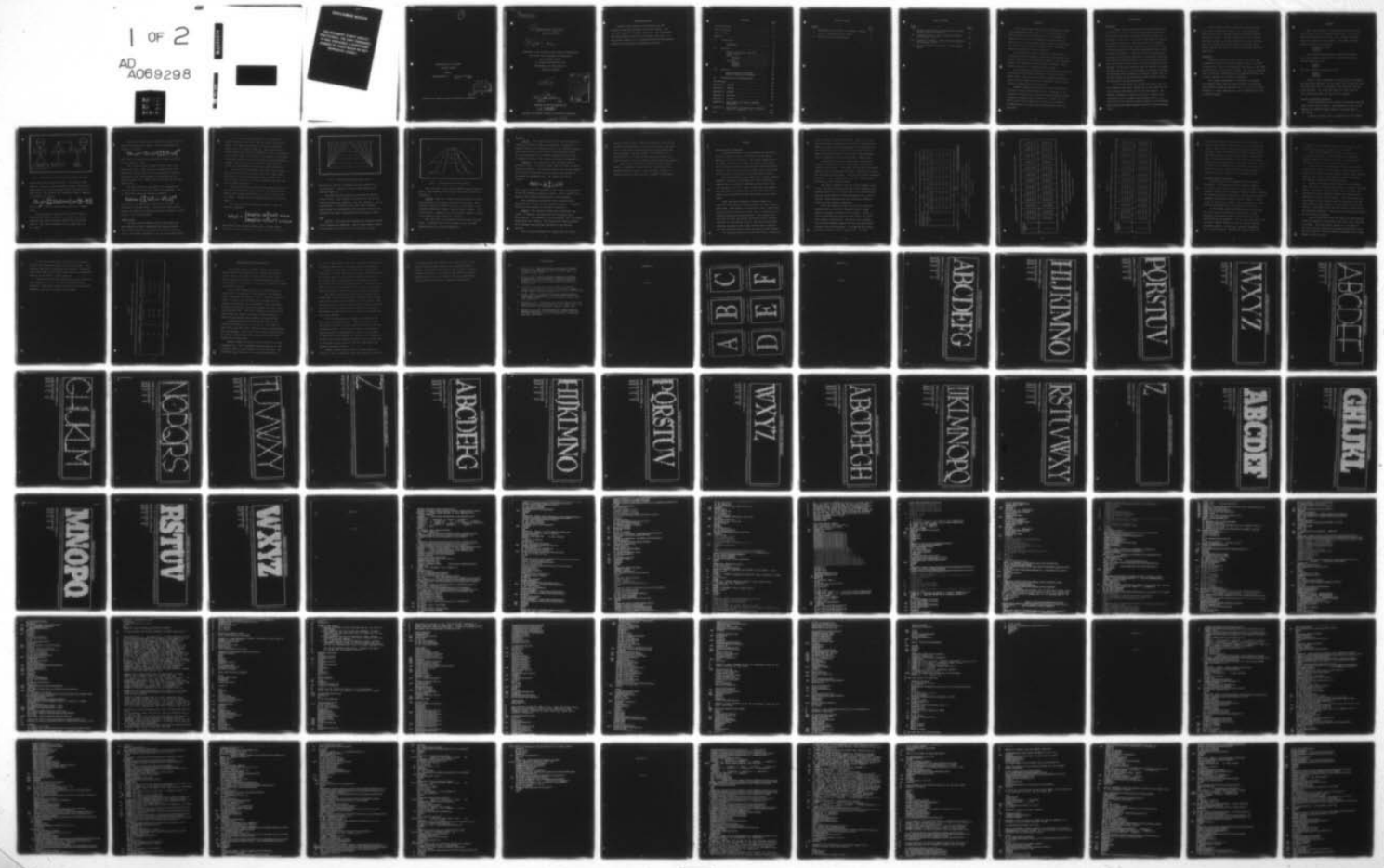
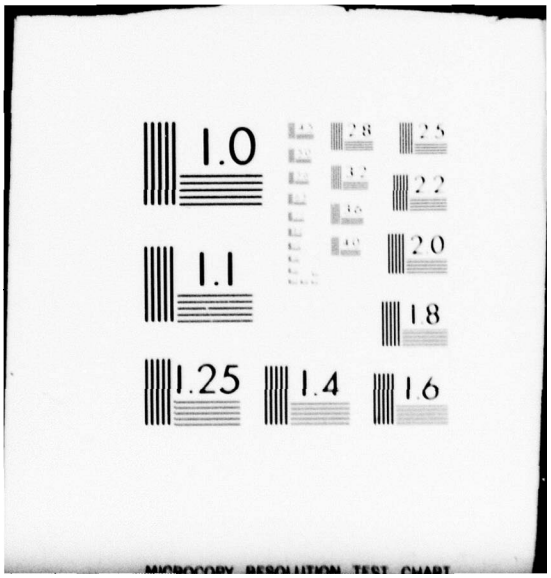
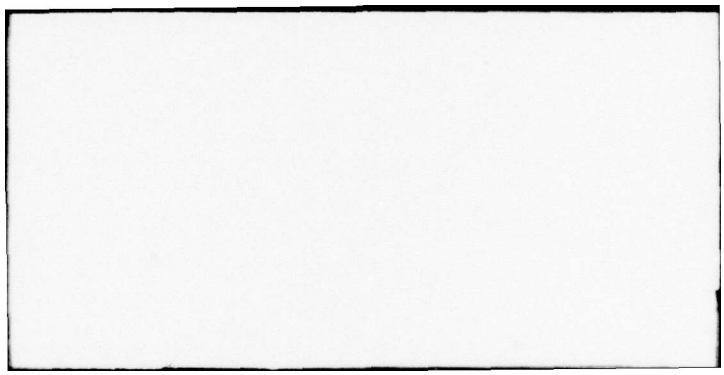
AD-A069 298 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 9/2
SEGMENTATION OF TOUCHING ENGLISH LETTERS.(U)
MAR 79 R E BENTKOWSKI
UNCLASSIFIED AFIT/GE/EE/79-1 NL




1.0
1.1
1.25
1.4
1.6
2.2
2.0
1.8
MICROCOPY RESOLUTION TEST CHART



AFIT/GE/EE/79-1

(1)

SEGMENTATION OF TOUCHING ENGLISH LETTERS

THESIS

AFIT/GE/EE/79-1 Roy E. Bentkowski
2nd Lt USAF

Approved for public release; distribution unlimited

(14) AFIT/GE/EE/79-1

(6) SEGMENTATION OF TOUCHING ENGLISH LETTERS.

(9) Master's THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air Training Command

in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

(12) 189 p.

(10) by Roy Edward Bentkowski, B.S.
2nd Lt USAF
Graduate Electrical Engineering

(11) March 1979

Acknowledgements

I wish to thank Captain Troy Sponaugle for the original recognition programs and the data files. I am also very grateful to Captain John Leary. His familiarity with the computer programs was an immense help to me. A very special thanks to Dr. Matthew Kabrisky for his valuable guidance and encouragement throughout my thesis.

Contents

List of Figures

List of Tables

Abstract

This paper examines the problem of building a machine to read uncontrolled type fonts set with essentially no space between letters (within words). The consequence of this type of data, which represents the usual format of printed text, is that the data vectors produced by the optical scanner contain multiple letters and/or fragments of letters that cannot be easily separated.

An algorithm based on a variant of running cross-correlation between prototype letters and successively "windowed" fragments of the sentence is employed. The algorithm computes the Euclidean distance between prototypes and the sentence fragment in a filtered Fourier domain.

It is shown that appropriate normalization and windowing techniques allow perfect recognition of touching letters within words. This occurs even when no apriori knowledge of letter location within the word is available, provided that suitable prototypes can be established.

Multiple alphabet prototypes were then built and used to examine widely differing type fonts. Techniques to set acceptance thresholds were evaluated and the behavior of the resulting recognition system tabulated. A number of false triggers did occur in this case and these were discussed. Recommendations for further improvements in the system are suggested.

Introduction

Background

A reading machine (to read uncontrolled, linotype-set fonts) based on correlation mathematics (Refs 3 and 4) has never been successfully operated, apparently because of the nature of the noise in the data stream that such systems are required to process. The noise arises from multiple mechanisms, including shape variations, and has never been well defined because the processes producing it seem to have no useful description. Empirical data for this noise does not seem to exist in the literature. Recognition algorithms are unable to reliably operate a threshold detection machine, even for isolated letters where the best prototype has been employed. In the more realistic case of touching (or nearly touching) letters, as set by linotype or equivalent machines, an additional noise consequent to multiple complete and incomplete signals in the observation window further reduces the accuracy of all known algorithms.

In a previous thesis by Captain Paul Kabler (Ref 2), 150 alphabets with widely varying fonts were digitized, and these serve as the basis data set for this thesis. The Fourier transforms of these vectors were obtained in the unpublished work of Captain Troy Sponaugle (Ref 5), and were used to test various isolated letters in a correlation-type recognition system against the kinds of noise mentioned above.

Captain Sponaugle then constructed various data sentences from the two-space letter data. These sentences were each made from a distinct alphabet with fixed spacing between the letters. Further improvements of the computer algorithms enabled reading the sentence by passing a window across the sentence by single element increments. This program and the data files are the basic research tools for this thesis.

Objectives

The major objective of this thesis is to solve the segmentation problem of touching English letters. A method of shaping the windows to de-emphasize the energy in the right and left edges of the letters is used. This method eliminates some of the noise introduced by the neighboring letters, and in fact, produces more reliable recognition.

The computer performance is evaluated by comparing the computer output with the actual content of the window; this enables acceptance thresholds to be set for each letter. Therefore, a simple decision rule can be implemented in this recognition system.

Approach

This section begins by summarizing the logic executed by the actual pattern recognition algorithm. It then continues by discussing the various window shaping functions that can be applied by the programs. The types of window shaping functions used include:

Unshaped (or Full)
Trapezoid
Gaussian

This section concludes with a summary of the data files and computer programs (Ref 5) used in this thesis. The two data files are:

LETTERS
SENDATA

The three computer programs used are:

MAKFLTR
PROTOBLD
CMPRSEN

Figure 1 gives a flowchart of the way that the programs and data files interact. In the figure, the circles represent the data files and the rectangular boxes represent the computer programs. The triangular boxes are files built by the indicated programs for use in later programs.

Pattern Recognition Algorithm

A brief description of the pattern recognition algorithm used in this thesis follows. This algorithm is used in the MAKFLTR and CMPRSEN program, which is discussed later in this thesis.

A window is brought in as a digitized 32 x 32 element

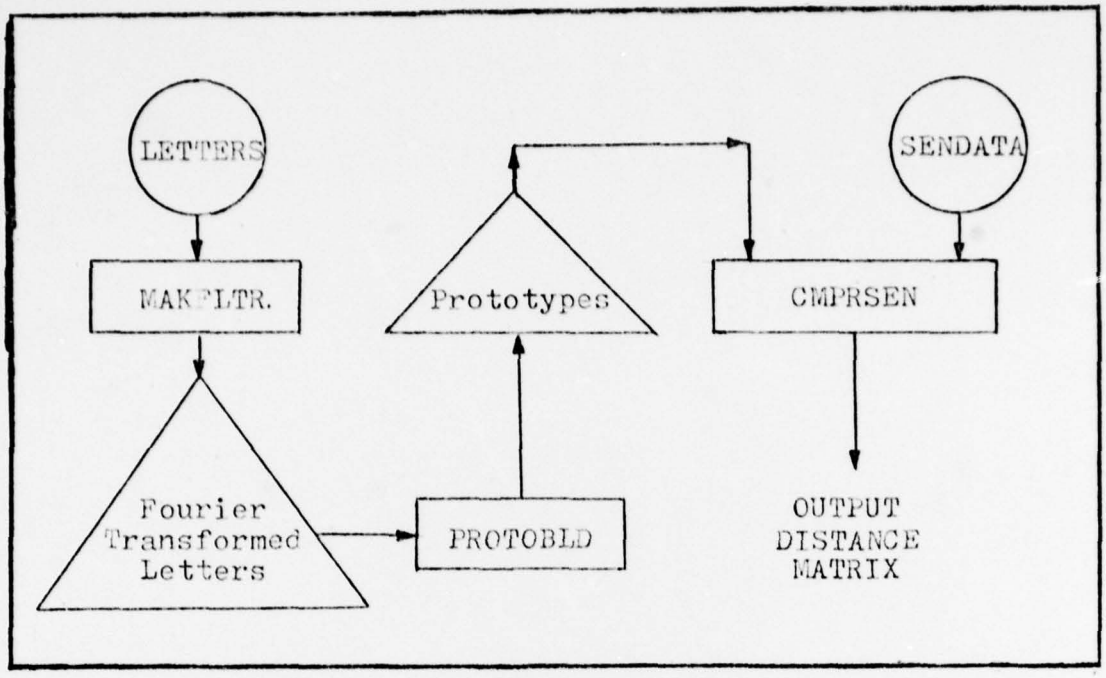


Fig 1. Interaction of Data Files and Computer Programs

array, consisting of zeros and ones. This array is then input to a discrete Fourier transform routine, FOURT (Ref 1), which obtains the two-dimensional Fourier transform of the input array. The discrete two-dimensional Fourier transform is calculated as follows:

$$O(r,s)=\sum_{k=1}^{K}\sum_{l=1}^{L} I(k,l)\exp\left[-j2\pi\left(\frac{kr}{K}+\frac{ls}{L}\right)\right]$$

where $j=\sqrt{-1}$, I(k,l) = input array, and O(r,s) = output array.

The array that is output from the FOURT routine is then further processed. First it is filtered in order to eliminate the spatial harmonics greater than the fourth (Ref 6). This allows reduction of the signal space to 81 - space.

The array is then energy normalized to insure that the energy of each vector is unity. Energy normalization is computed as follows:

$$N(r,s) = O(r,s) / \left[\sum_{r=1}^{2} \sum_{s=1}^{2} O^2(r,s) \right]^{1/2}$$

where O(r,s) is output array from FOURT routine and N(r,s) is the normalized array.

The normalized array is then reordered into a one-dimensional array. This new array contains the DC term followed by the real part of the first harmonic, the imaginary part of the first harmonic, the real part of the second harmonic, etc. This is the same format used for all data vectors.

The next step in the algorithm is to calculate the Euclidean distance between the input and a prototype. This is simply done using the following equation:

$$\text{Distance} = \left[\sum_{i=1}^{81} [W^2(i) - P^2(i)] \right]^{1/2}$$

where W(i) = one-dimensional window array and P(i) = one-dimensional prototype array. Thus when the input array is exactly the same as the prototype array the distance is equal to zero - or a perfect match.

Window Shapes

The three window functions used in this thesis are the unshaped (or full), trapezoidal and Gaussian shapes. These various functions are applied to windows in both the MAKFLTR and CMPRSEN programs (see the Program section below).

Each of these window functions are calculated to obtain 32 normalized values, with the highest value being unity. The values are put into an array such that the unity values are centered within it. Then each of the 32 columns within the window are multiplied by the corresponding array value. This process causes a de-emphasis of the window edges, thus reducing the noise introduced by the neighboring letters. This process also reduces the signal but not enough to significantly change the recognition distances.

In the case of the unshaped (or full) window function the above described array contains all ones. This allows the window to be read in directly, with no change made. It is used only to see the effect the neighboring letters have on recognition distances.

In this thesis the trapezoid plateau length was varied in order to study window shape affects. The various values used for the plateau length were 1, 5, 10, 15, 20, 25, and 30 columns. Examples of these window functions are given in Figure 2.

The Gaussian window function was obtained using the following equation:

$$W(x) = \begin{cases} \exp[-(x-16)^2/2s^2] & x = 1,2,\cdots,16 \\ \exp[-(x-17)^2/2s^2] & x = 17,18,\cdots,32 \end{cases}$$

where W(x) is the Gaussian window array, x is the column position along the window, and s is the standard deviation.

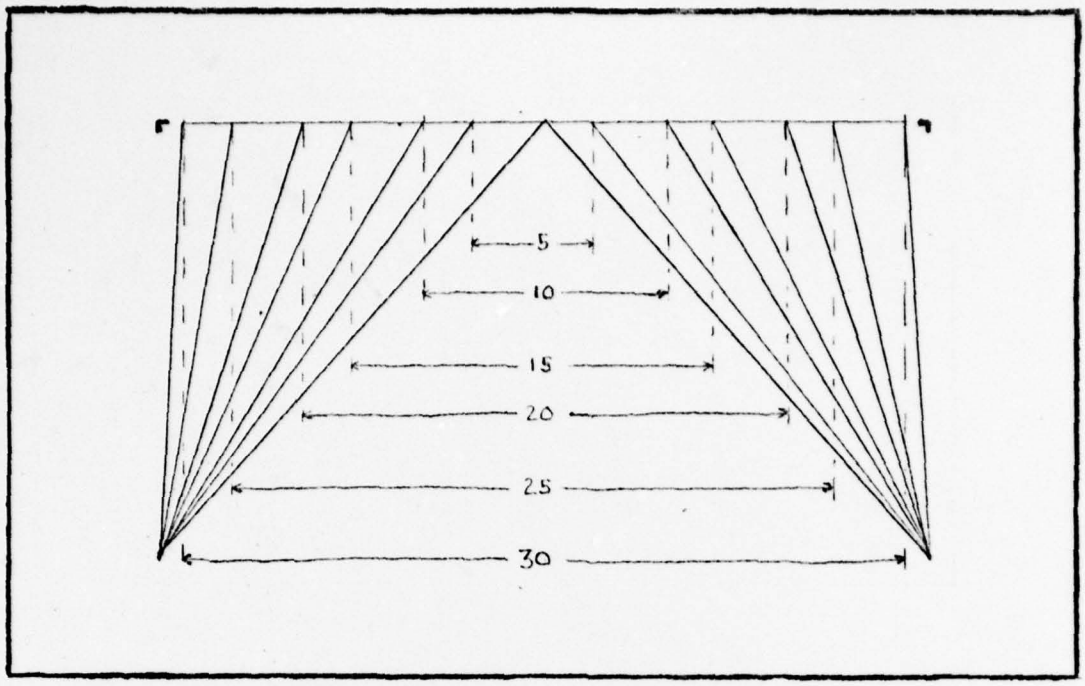


Fig 2. Various Trapezoidal Window Shapes

The range of standard deviations used were values of 2, 5, and 10 columns. Examples of these window functions are given in Figure 3.

The above described process is executed in the subroutine TRPMLT. This routine is listed in the MAKFLTR and CMPRSEN programs (see Appendix C, pg. 69; Appendix E, pg. 101 respectively). It should be noted that these listings contain only the Gaussian logic, but any type window can be implemented by simply substituting the new function logic.

Data

LETTERS. This data file contains the digitized letters of 150 various font alphabets. Each of these stored letters are isolated and centered in a 32 x 32 element window.

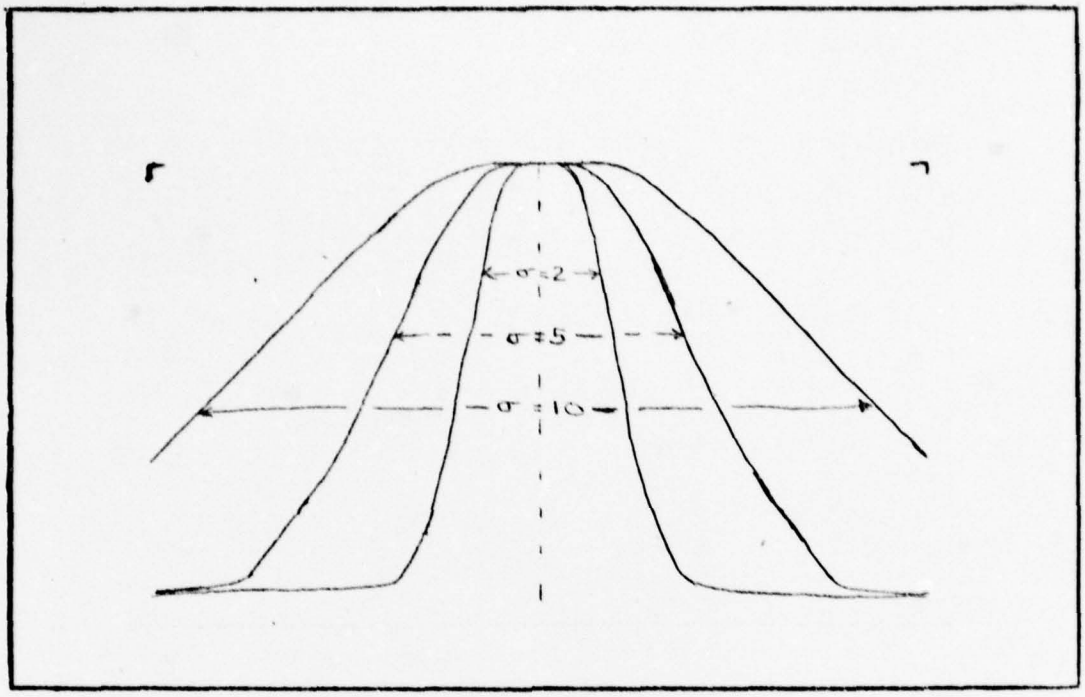


Fig 3 Various Gaussian Window Shapes

This data file is used by the MAKFLTR program discussed below. Some examples of the data in this file are shown in Appendix A.

SENDATA. This file contains various numbered sentences. Each sentence is 133 columns long and 32 rows high. These sentences consist of letters in a line, in the manner of an ordinary sentence but with no spacing between letters. Each letter within a sentence comes from a single alphabet so there is no mixing of fonts within the sentence.

This data file is used by the CMPRSEN program discussed below. A complete list of the sentences used, and their characteristics are shown in Appendix B.

Programs

MAKFLTR. This program produces the Fourier transformed letters necessary to build the prototypes. It applies a shaped window to each isolated letter in the data file LETTERS. This shaped letter is then Fourier transformed and stored in the format used by the PROTOBLD program discussed below. A complete listing of this program is included in Appendix C.

PROTOBLD. This program uses the file built by the MAKFLTR program. It uses the shaped Fourier letters to build average prototypes. The averaging process used is the summing of the Fourier components for each letter and dividing by the number of alphabets used. The formula used is as follows:

$$P(K) = \frac{1}{N} \sum_{i=1}^{N} L_i(K)$$

Where $L_i(K)$ is the K^{th} component of the Fourier representation of the i^{th} alphabet-version of a letter. $P(K)$ is the K^{th} component of the Fourier representation of the prototype.

This program also permits the user to specify which alphabets are to be used in the averaging process. A complete listing of this program is included in Appendix D.

CMPRSEN. This program contains the recognition algorithm. It requires, as input, both the prototypes and a data sentence. The sentence is read by passing a 32 x 32 element window along the sentence. The input at each window position is run through the algorithms discussed in the previous sections.

These Euclidean distances are output from the CMPRSEN

program in two formats. The first format is a list of the distance each prototype is from each window along the sentence. This allows easy observation of each distance between prototypes and the contents of the window. The second format is a listing of the best five choices (i.e. lowest distances) for each window position along the sentence.

There is an additional capability of using either complex or amplitude Fourier spectra. However the same Euclidean distance logic is used for either spectra. A complete listing of the program is included in Appendix E.

Results

Single Alphabet Prototypes

Two measurements are used to evaluate the performance of the recognition system. The first is the Euclidean distance to the correct choice. The second measurement is the figure of merit (FOM), which is defined as the ratio of the distances from the second choice to the first choice. Therefore, the FOM has a range of values greater than unity as long as the first choice is also the correct choice.

The first prototypes were built from a single, unshaped alphabet, identified as alphabet #10. This is the same alphabet used to construct sentences numbered one to four (see Appendix B, pg. 26). Thus we have perfect prototypes for each letter in alphabet #10; in the sense that the sentences are constructed from the same letters as the prototypes.

Using the above prototypes, sentence number one ("ABCDEFG") was read in with the unshaped (or full) function being used. This allowed a standard for further comparisons. The only noise in the data, in this case, is provided by the adjacent letters in the scan windows. This sentence was also read in using the various trapezoidal-shaped function described in the previous section. The lowest recognition distances and highest FOM to the correct choices occurred when the trapezoid plateau length was equal to five columns.

New prototypes were then constructed with a trapezoidal

shape of plateau length equal to five columns and various trapezoidal shaped functions were applied to read sentence number one using these new prototypes. This time the best recognition distances occurred for trapezoidal function shape with plateau length of one column. These empirical results suggested a function shape with a sharper peak.

A Gaussian function was chosen next. The various Gaussian prototypes were each run through the different Gaussian window shapes. From these runs the best recognition occurred when the prototypes and window function had a standard deviation of two columns. The above results are summarized in Table I. Note especially the very low, virtually zero, distances to the correct prototype.

The data in Appendix F are the actual computer output of the five lowest distances for each window position along sentence number one. The three outputs are for both prototype and window standard deviations equal to 2, 5, and 10 columns. These data show how the recognition distances depend upon the standard deviations.

These prototypes are then run along the sentences numbered one to four with a Gaussian function shaped window of standard deviation of two columns. Tables II and III summarize the output from the complex and amplitude spectra, respectively. The tables give a listing of the lowest distance achieved by each prototype filter on each of the four sentences. It shows how the correct prototype distances fall to nearly zero. It should

TABLE I

Distance and Figure of Merit Data for Various Window Shapes on Sentence 1

Prototypes	A10				A10/T5		A10/G2	
Window	FULL		T5		T1		G2	
Letter	DX	FOM	DX	FOM	DX	FOM	DX	FOM
B	.786	1.085	.376	1.495	.353	1.530	.000	und
C	1.053	1.174	.536	1.664	.507	1.712	.000	und
D	.798	1.189	.408	1.716	.366	1.861	.000	und
E	.932	1.048	.524	1.389	.521	1.376	.001	413.
F	.792	1.229	.450	1.798	.443	1.725	.004	103.
G	.487	1.608	.303	2.327	.248	2.605	.000	und

KEY

DX→distance to first choice FOM = Figure of Merit = $\frac{\text{Distance to second choice}}{\text{Distance to first choice}}$

An→A: Alphabet n = alphabet number used

Tm→T: Trapezoid m = length of trapezoid plateau

Gj→G: Gaussian j = standard deviation of Gaussian window

und = undefined

TABLE II

Complex Spectrum - Lowest Distance Achieved by Prototype Filters

Sentence Number	A	B	C	D	E	F	G	H	I	J	K	L	M
1	.638*	.000	.000	.000	.001	.004	.000	.541	.219	.330	.564	.763	.710
2	.590	.793	.519	.305	.672	.514	.392	.942*	.066	.000	.000	.824†	.000
3	.475	.556	.556	.652	.506	.434	.669	.637	.172	.366	.500	.761	.551
4	.490	.846	.701	.284	.777	.828	.635	.861	.687	.759	.428	.755	.402

	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	.789	.431	.412	.723	.506	.572	.183	.624	.439	.828	.628	.649	.699
2	.000	.000	.738	.741	.603	.635	.223	.689	.502	.735	.317	.515	.683
3	.550	.671	.421*	.000	.000	.014	.000	.000	.000	.366	.556	.409	.737
4	.317	.554	.876	.827	.830	.621	.710	.736	.295	.000	.000	.000	.000

Prototypes - Alphabet #10, Gaussian Shape (Standard Deviation = 2)

Window - Gaussian Shape (Standard Deviation = 2)

* Letter never centered in window

† Bad data letter

TABLE III

Amplitude Spectrum - Lowest Distance Achieved by Prototype Filters

Sentence Number	A	B	C	D	E	F	G	H	I	J	K	L	M
1	.246*	.000	.000	.000	.000	.002	.000	.355	.116	.131	.289	.416	.206
2	.204	.524	.207	.187	.467	.391	.211	.500*	.051	.000	.000	.395†	.000
3	.204	.366	.330	.244	.306	.259	.363	.358	.120	.105	.206	.352	.180
4	.222	.586	.112	.197	.536	.501	.298	.485	.399	.348	.212	.367	.164

	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	.358	.223	.251	.346	.309	.252	.092	.244	.258	.329	.290	.221	.420
2	.000	.000	.485	.275	.343	.461	.057	.330	.181	.336	.182	.230	.399
3	.326	.345	.213*	.000	.000	.012	.000	.000	.000	.249	.282	.209	.411
4	.272	.294	.554	.275	.439	.360	.345	.286	.133	.000	.000	.000	.000

Prototypes - Alphabet #10, Gaussian Shape (Standard Deviation = 2)

Window - Gaussian Shape (Standard Deviation = 2)

* Letter never centered in window

† Bad data letter

be noted that the way the sentence is read in causes the first letter of each sentence never to be centered in the window. This is the reason that the letters A, H and P do not drop to zero. This was not corrected in this project (by adding leading columns of zeros to the sentences) because of time considerations. The letter L, in sentence number two is incomplete because when the sentence was built the lower bar was omitted (see Appendix B, pg. 28). These data errors can be easily corrected once the sentence building program is restored to the computer files.

Multiple Alphabet Prototypes

The next step was to introduce multiple alphabet prototypes. The alphabets used were Kabler's numbers 10, 11, 12, 13, 14 and 15 (Ref 2). These are some of the alphabets (number 15 was excluded) that were used to construct the sentence data (see Appendix B). Note that with multiple alphabets an additional source of noise, really a "shape" noise, is added to the problem. As a consequence one would expect the performance of the system to be degraded.

In the single alphabet prototype case, it was found that the Gaussian shaped window gave perfect results. Therefore, the various Gaussian shaped prototypes were built from the six alphabets. Each of these prototypes were run along sentence number one with the various Gaussian functions applied. In this case, the best recognition (i.e. lowest distances, highest FOM) occurred when the

prototypes had a standard deviation of ten columns and the window had a standard deviation of five columns.

These best prototypes were then run along all 22 of the data sentences. Both the complex and amplitude spectra output was obtained. However, as expected, the distances in this case did not drop to zero.

Some examples of the raw data output from the above discussion is contained in Appendix G. These data are the lowest five distances for each window position along sentences numbered one to four. This Appendix contains the output for both the complex and amplitude spectra.

Acceptance thresholds were then set for each letter. This would allow a simple decision scheme to make the detection choice. Therefore, for each window, where the letter was relatively centered within the window, the distances were recorded for each of the 26 letters. This was repeated for each letter in every sentence until five distances (one for each alphabet), were obtained for each letter, and for each spectra. The highest number for each letter was then set as the acceptance threshold. Therefore, all letters were detailed by the threshold mechanism.

Using the Euclidean distances output for each prototype along the sentences, every number below that letter threshold was marked. When both spectra agreed (i.e. triggered at the same sentence position) the program declared that letter was in the window. But when this was done there were a large number of false triggers.

It was then observed that the letters from alphabet #11 were determining thresholds in most of the cases. These sentences were made of atypically thin letters. Thresholds were reset omitting this alphabet and there was a substantial decrease in the number of false triggers.

A summary of these false triggers is given in Table IV. Five letters (I, J, O, Q, W) were consistently falsely triggered. Other false triggers did occur for the different alphabets, and these are entered in the table.

TABLE IV

Multiple Alphabet Prototypes - False Trigger Chart

Alphabet Number	Sentence Numbers	False Trigger Letters
10	1 - 4	A, B, C, N, R, T
12	10 - 13	C, D, H, L, U
13	14 - 17	D, H, R
14	18 - 22	A, B, E, L, R, S

Note: Letters I, J, O, Q, W missed in all alphabets

Conclusions and Recommendations

The single alphabet prototype results showed that a perhaps interesting fragment of the segmentation problem can be solved. Since the distances fall to nearly zero only on the correct letters, a simple threshold detector can be implemented. The detector can be used on either the complex or amplitude spectra.

The multiple alphabet prototype results turned out fairly well. The problem of false triggers that occurred could be attacked in a number of ways. First, and most obvious, is based on the fact that the prototypes were an average of the letters in six alphabets. These alphabets included an atypically thin font (alphabet #11) and a thick font (alphabet #14). This wide variation in font might very well account for many of the false triggers obtained. A method of ameliorating this problem is suggested below.

It would be possible to add special logic for the troublesome letters (i.e. I, J, O, Q, W). An example of this occurs when the B and I filters trigger at approximately the same location. Since the I filter would normally trigger in multiple neighboring windows, the special logic would trigger the B filter only.

Another example of special logic could be used in the following case. The recognition system triggers B, I, and E filters, within a small number of window locations. The additional logic would use the average column lengths of

B, I and E, taking into account the fact that an I could never fit between these letters, and override the I triggering.

There are other types of special logic that might be implemented. Perhaps logic based on English constraints could be included such as the rule that U always follows a Q. It is also possible to adaptively vary the window size and re-evaluate the decision computation when an ambiguity arises. However, to keep the generality of this process it is to be hoped the amount of special logic can be kept to a minimum.

The obvious next step in this project would be to go through the 150 alphabets and divide the fonts into separate categories according to the thickness of line strokes. Then prototypes for each letter could be built in these categories and tested against various sentences within the category. New thresholds could then be set as discussed previously and the results studied.

It is also necessary to develop a font detector for the overall recognition scheme. This device would enable the machine to decide the most suitable prototype to use for any input sentence. It is hoped the number of prototypes required would be considerably less than the total number of fonts read by the machine. For instance, by observation of the first Fourier transformed window of the sentence, the DC term could be used to decide the most appropriate line thickness prototype to use.

Another recommendation would be to investigate the behavior of this recognition system with arbitrary sequences

of letters within the sentence, since this thesis is concerned with the noise introduced by neighboring letters. However, the sentence data used in this thesis is of a set sequence of letters. For example, the letter C is always between letters B and D. Therefore the many combinations at different letter triplets should be investigated.

Bibliography

1. Haller, Mark. The Cooley-Tukey Fast Fourier Transform in USASI Basic Fortran. A Computer Program for the CDC 6600. Wright-Patterson Air Force Base, Ohio: ASD Computer Center, 1972.

2. Kabler, Paul W. Fourier Spatial Frequency Components as Descriptors for Recognition of Variable Font Typescript. Unpublished MS Thesis, Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, December 1975.

3. McLachlan, Dan, Jr. "The Role of Optics in Applying Correlation Functions To Pattern Recognition", Journal of the Optical Society of America, 52: 454-459 (April 1962).

4. Pirich, Andrew R. A Survey of Optical Character Recognition (OCR) Technology. Technical Report, Griffis Air Force Base, New York: Rome Air Development Center, March 1973 (AD909342L).

5. Sponaugle, Troy. Electrical Engineering Student (personal communication) Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, January - March 1979.

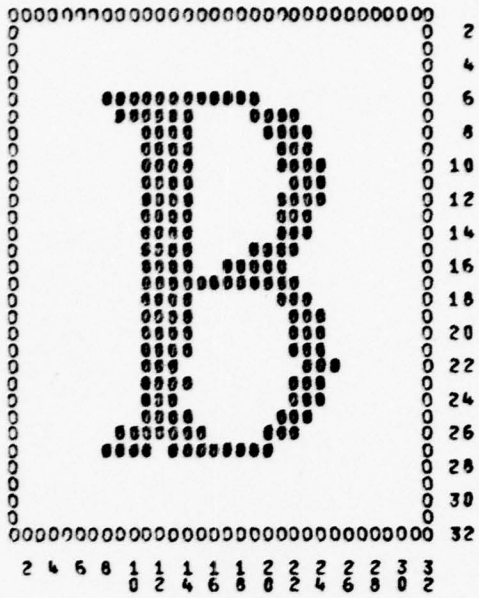
6. Tallman, O. H. The Classification of Visual Images by Spatial Filtering. PHD Dissertation, Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, June 1969 (AD858866).

Appendix A

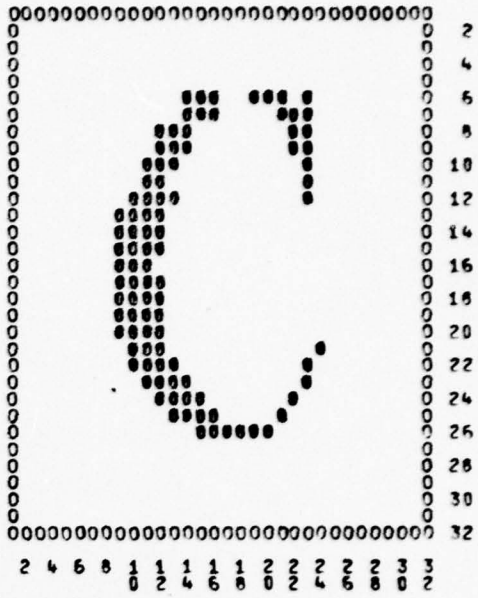
LETTERS

*** ALPHABET 10, LETTER 1 ***

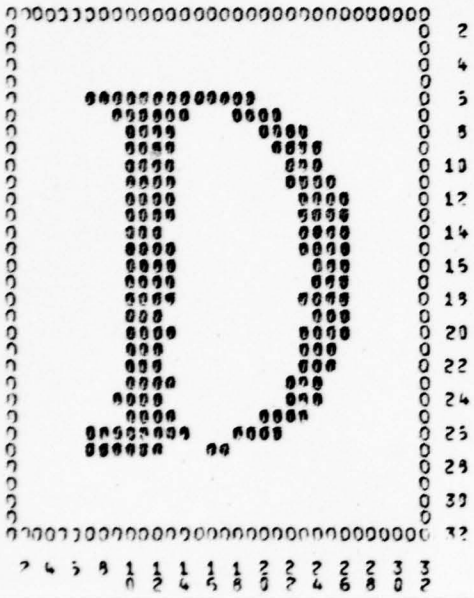
*** ALPHABET 10, LETTER 2 ***



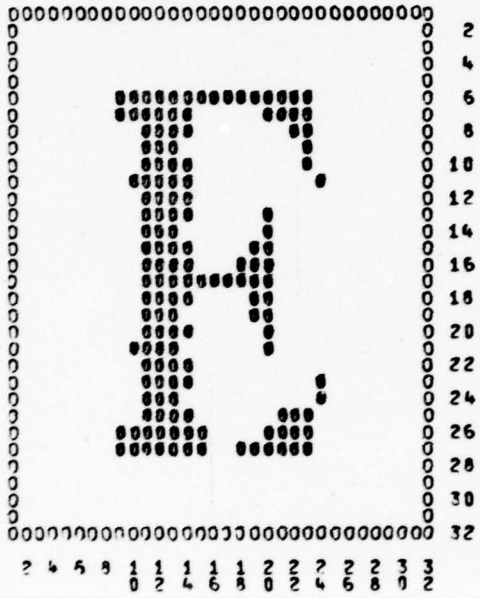
*** ALPHABET 10, LETTER 3 ***



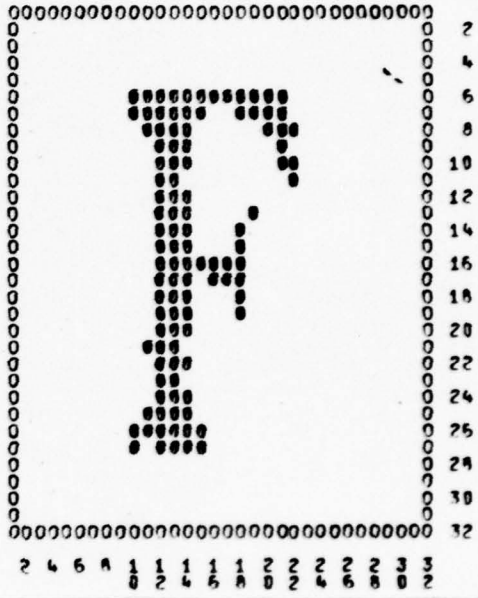
*** ALPHABET 10, LETTER 4 ***



*** ALPHABET 10, LETTER 5 ***



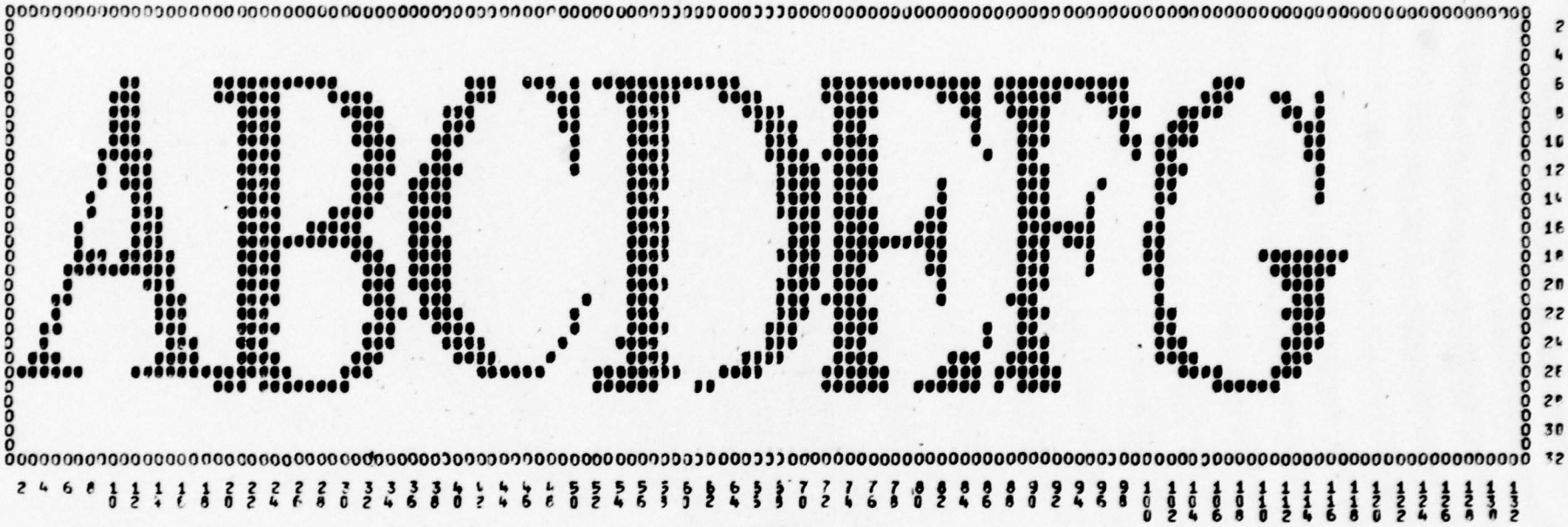
*** ALPHABET 10, LETTER 6 ***



Appendix B

SENDATA

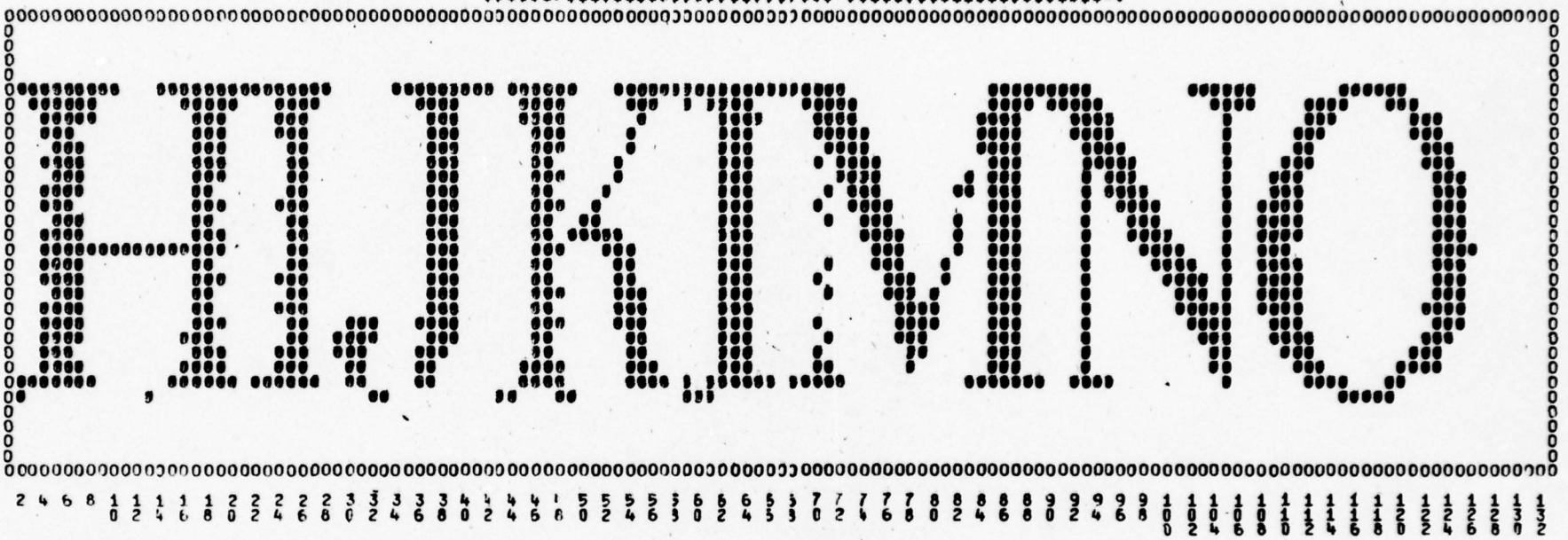
CHARACTERISTICS OF SENTENCE DATA: SENTENCE NUMBER 1



SENTENCE # 1 COLUMN 1 TO 133
LETTERS APPEARING IN THE SENTENCE ARE:

COLUMNS	WIDTH	ALPHABET	LETTER
2- 18	17	10	1
19- 35	17	10	2
36- 51	16	10	3
52- 71	20	10	4
72- 86	15	10	5
87- 99	13	10	6
100-117	18	10	7

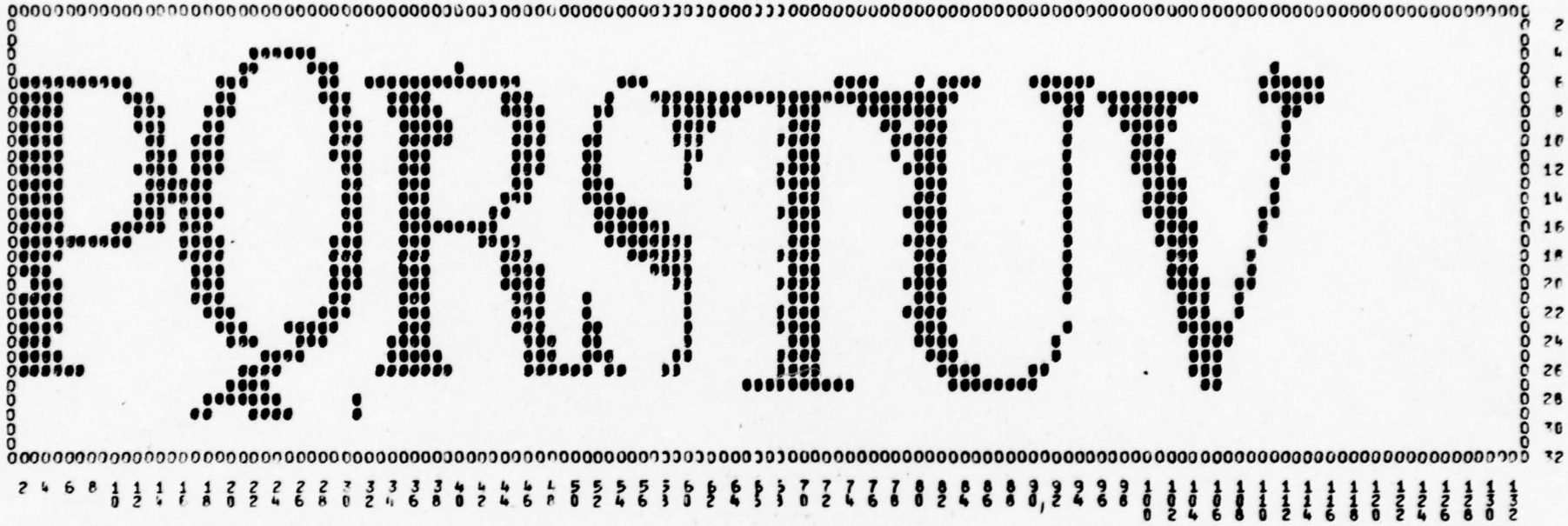
CHARACTERISTICS OF SENTENCE DATA: SENTENCE NUMBER 2



SENTENCE # 2 COLUMN 1 TO 133
LETTERS APPEARING IN THE SENTENCE ARE:

COLUMNS	WIDTH	ALPHABET	LETTER
2- 21	20	10	8
22- 28	7	10	9
29- 42	14	10	10
43- 59	17	10	11
60- 67	8	10	12
68- 89	22	10	13
90-107	18	10	14
109-126	19	10	15

CHARACTERISTICS OF SENTENCE DATA: SENTENCE NUMBER 3.

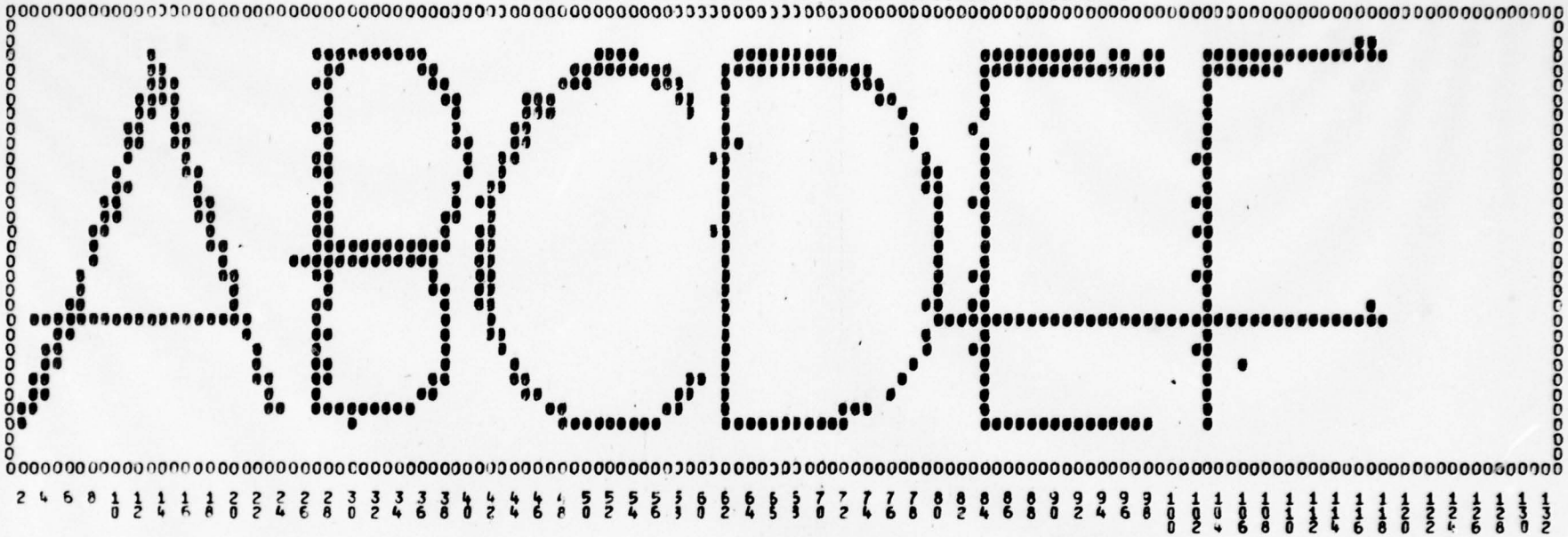


SENTENCE # 3 COLUMN 1 TO 133
LETTERS APPEARING IN THE SENTENCE ARE:

COLUMNS	WIDTH	ALPHABET	LETTER
2- 15	14	10	16
15- 31	15	10	17
32- 50	19	10	18
51- 60	10	10	19
61- 78	18	10	20
79- 95	17	10	21
96-115	20	10	22

LENGTHS OF SENTENCE AREA
SENTENCE # PER COLUMN THAT IS THIS SENTENCE AREA
COLUMNS WIDTH ALPHABET LETTER
2- 31 30 10 23
32- 51 20 10 24
52- 72 21 10 25
73- 89 17 10 26
"THE PROCESSING OF SENTENCE NUMBER 4 DATA SENTENCES, IS COMPLETED."

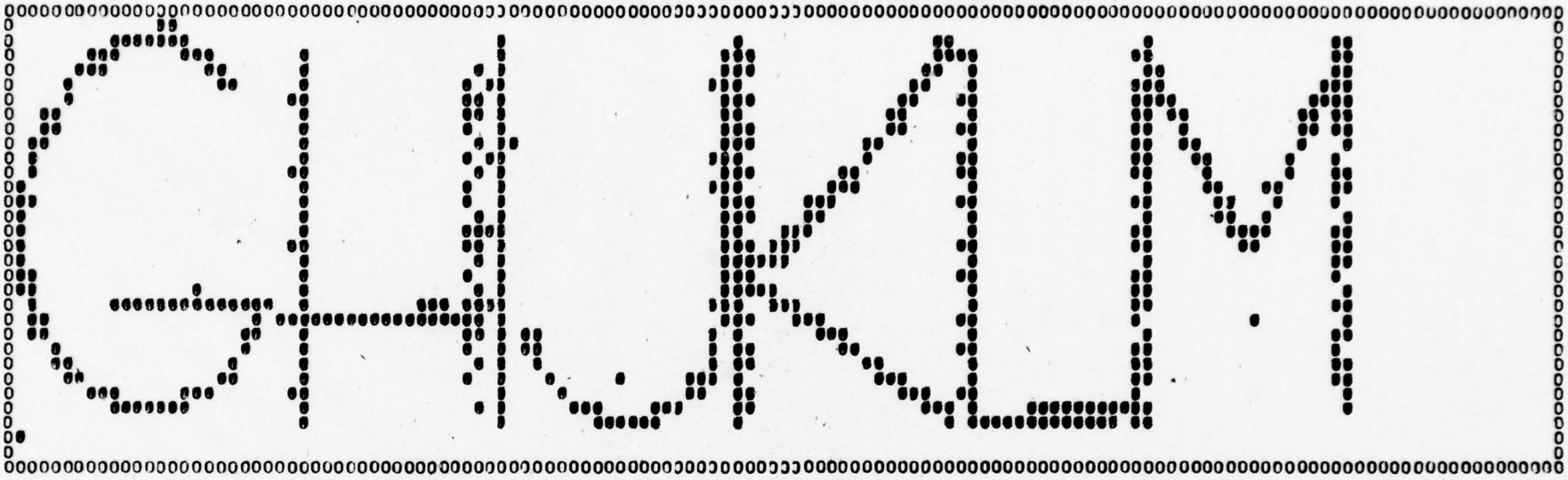
CHARACTERISTICS OF SENTENCE DATA: SENTENCE NUMBER 5



SENTENCE # 5 COLUMN 1 TO 133
LETTERS APPEARING IN THE SENTENCE ARE:

COLUMNS	WIDTH	ALPHABET	LETTER
2- 24	23	11	1
25- 40	16	11	2
41- 60	20	11	3
61- 80	20	11	4
81- 99	19	11	5
100-118	19	11	6

CHARACTERISTICS OF SENTENCE DATA: SENTENCE NUMBER 5.



SENTENCE # 5 COLUMN 1 TO 133
LETTERS APPEARING IN THE SENTENCE ARE:

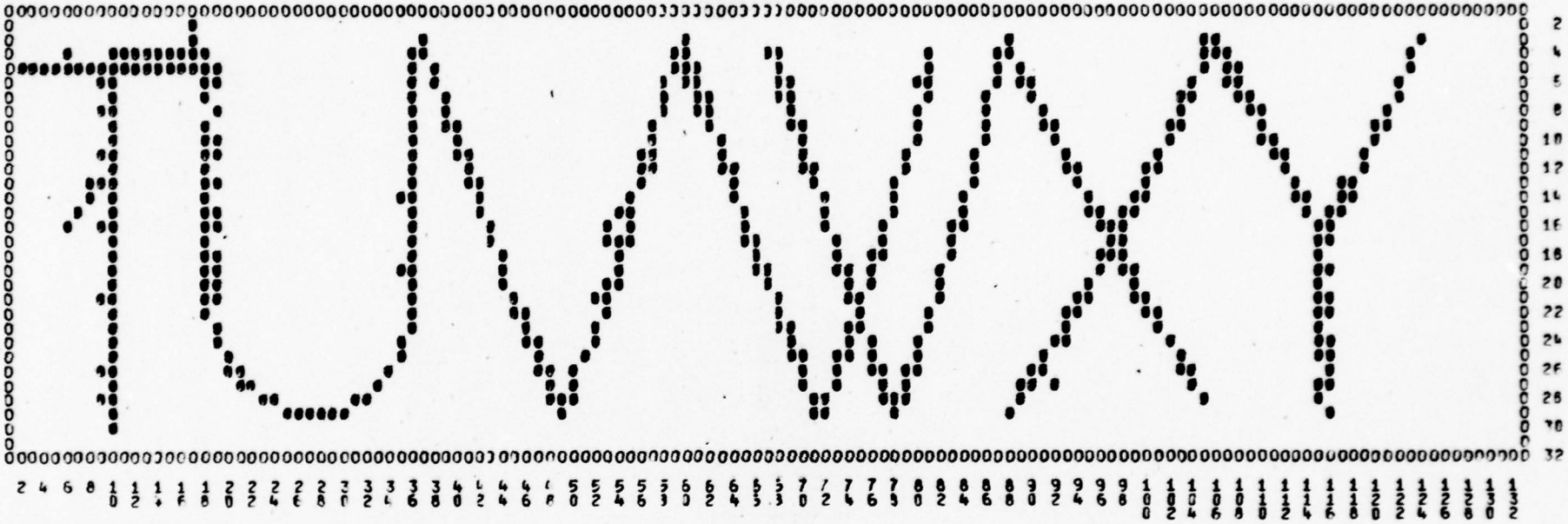
COLUMNS	WIDTH	ALPHABET	LETTER
2- 23	22	11	7
24- 41	18	11	8
42- 44	3	11	9
45- 62	18	11	10
63- 81	19	11	11
82- 96	15	11	12
97-115	19	11	13

SENTENCE # 7 CONTAINS IN THE SENTENCE ARE:
LETTERS APPEARING 1 TO 133

COLUMNS WITH ALPHABET LETTER	
2-- 20	13
21-- 42	22
43-- 59	17
60-- 82	23
83--102	20
103-120	16

14	11
15	11
16	11
17	11
18	11
19	11

CHARACTERISTICS OF SENTENCE DATA: SENTENCE NUMBER 8.



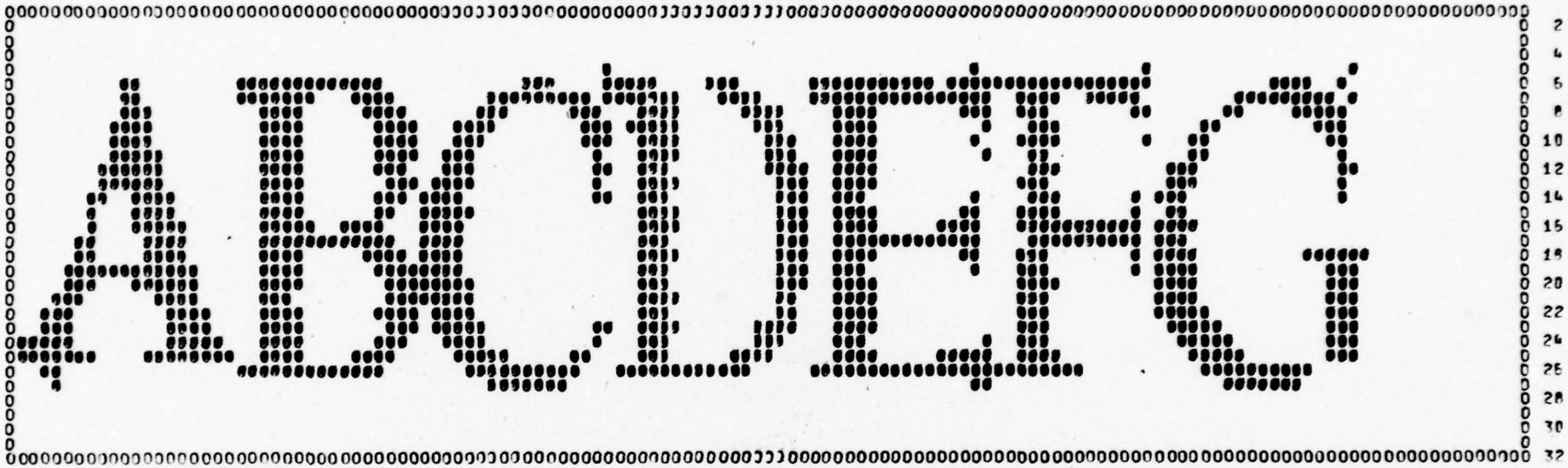
SENTENCE # 8 COLUMN 1 TO 133
LETTERS APPEARING IN THE SENTENCE ARE:

COLUMNS	WIDTH	ALPHABET	LETTER
2- 17	15	11	20
18- 36	19	11	21
37- 59	23	11	22
60- 87	28	11	23
88-105	18	11	24
105-124	19	11	25

SENTENCE # 3 COLUMN 1 TO 133
LETTERS APPEARING IN THE SENTENCE ARE:

COLUMNS	WIDTH	ALPHABET LETTER
2- 20	19	11 26

CHARACTERISTICS OF SENTENCE DATA! SENTENCE NUMBER 10.

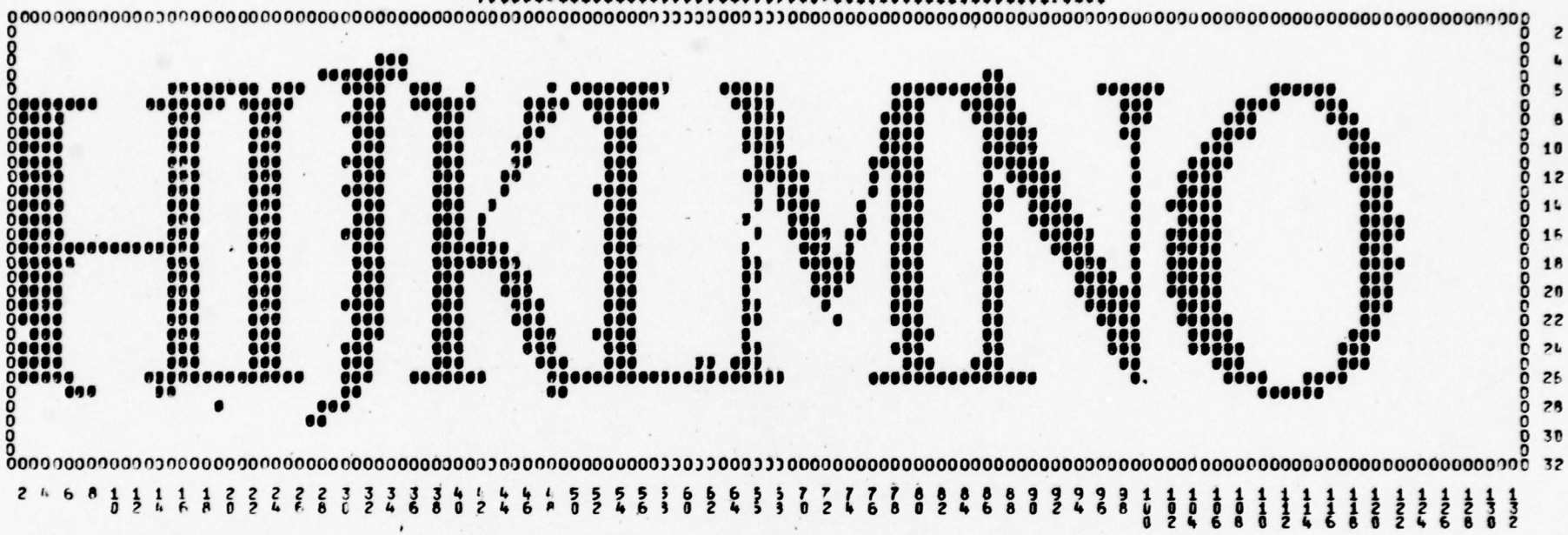


2 4 5 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80 82 84 86 88 90 92 94 96 98 100 102 104 106 108 110 112 114 116 118 120 122 124 126 128 130 132

SENTENCE # 10 COLUMN 1 TO 133
LETTERS APPEARING IN THE SENTENCE ARE!

COLUMNS	WIDTH	ALPHABET	LETTER
2- 20	19	12	1
21- 36	15	12	2
37- 53	17	12	3
54- 70	17	12	4
71- 85	15	12	5
87-100	14	12	6
101-119	19	12	7

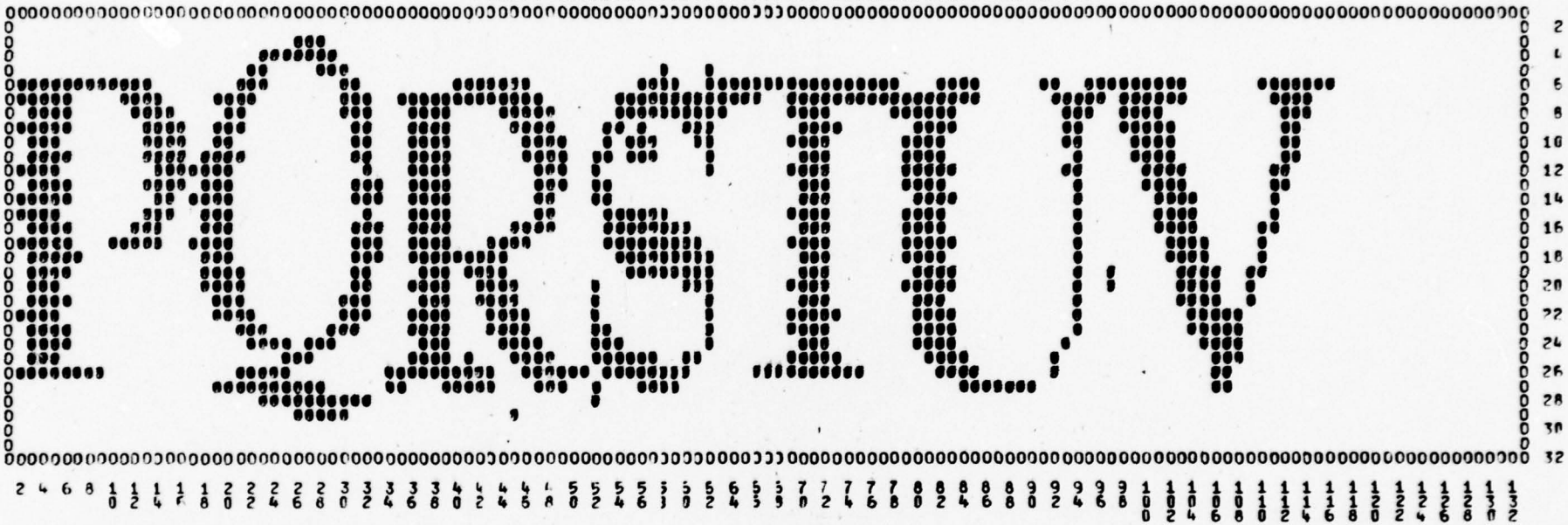
CHARACTERISTICS OF SENTENCE DATA: SENTENCE NUMBER 11.



SENTENCE # 11 COLUMN 1 TO 133
LETTERS APPEARING IN THE SENTENCE ARE:

COLUMNS	WIDTH	ALPHABET	LETTER
2- 19	18	12	8
20- 25	7	12	9
27- 35	9	12	10
36- 49	14	12	11
50- 62	13	12	12
63- 82	20	12	13
83-101	19	12	14
102-122	21	12	15

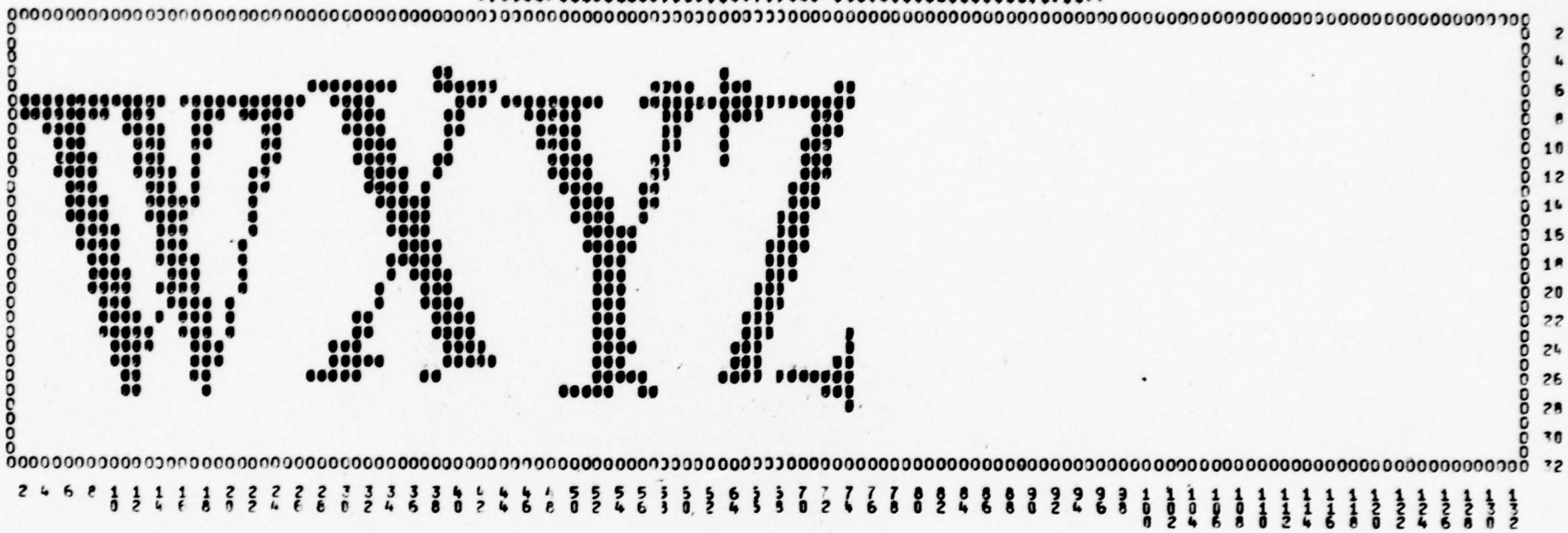
CHARACTERISTICS OF SENTENCE DATA: SENTENCE NUMBER 12.



SENTENCE # 12 COLUMN 1 TO 133
LETTERS APPEARING IN THE SENTENCE ARE:

COLUMNS	WIDTH	ALPHABET	LETTER
2- 16	15	12	16
17- 33	17	12	17
34- 51	18	12	18
52- 62	11	12	19
63- 78	16	12	20
79- 95	18	12	21
97-116	20	12	22

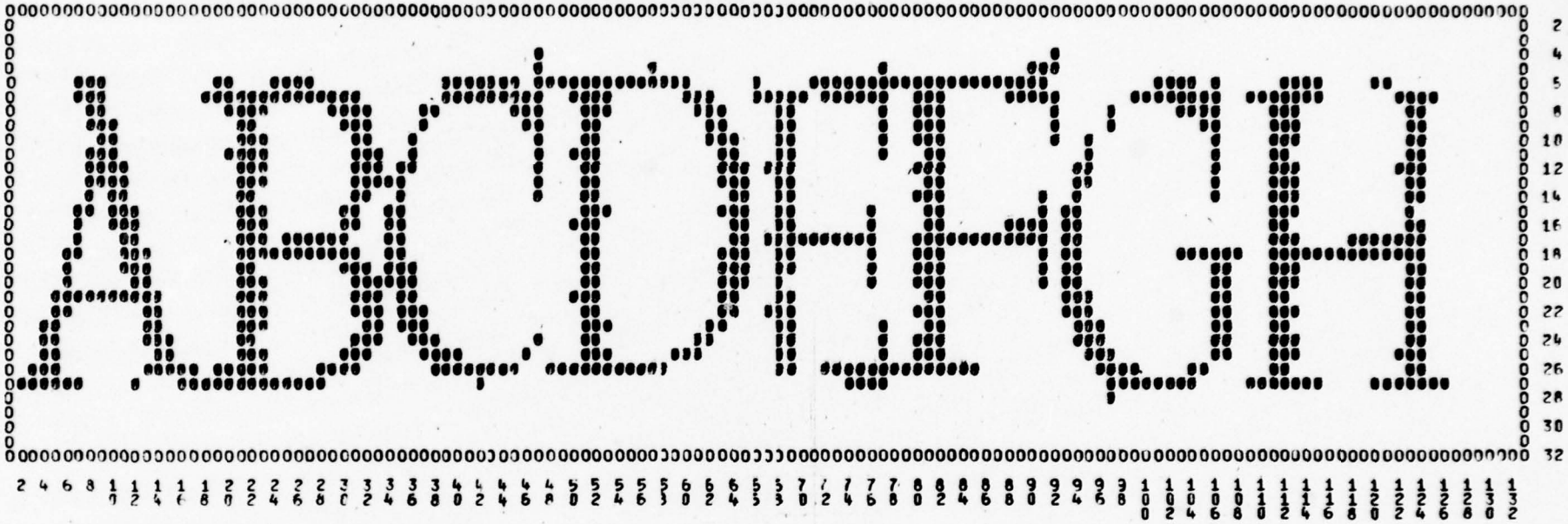
CHARACTERISTICS OF SENTENCE DATA: SENTENCE NUMBER 13.



SENTENCE # 13 COLUMN 1 TO 133
LETTERS APPEARING IN THE SENTENCE ARE:

COLUMNS	WIDTH	ALPHABET	LETTER
2- 26	25	12	23
27- 43	17	12	24
44- 62	19	12	25
63- 74	12	12	26

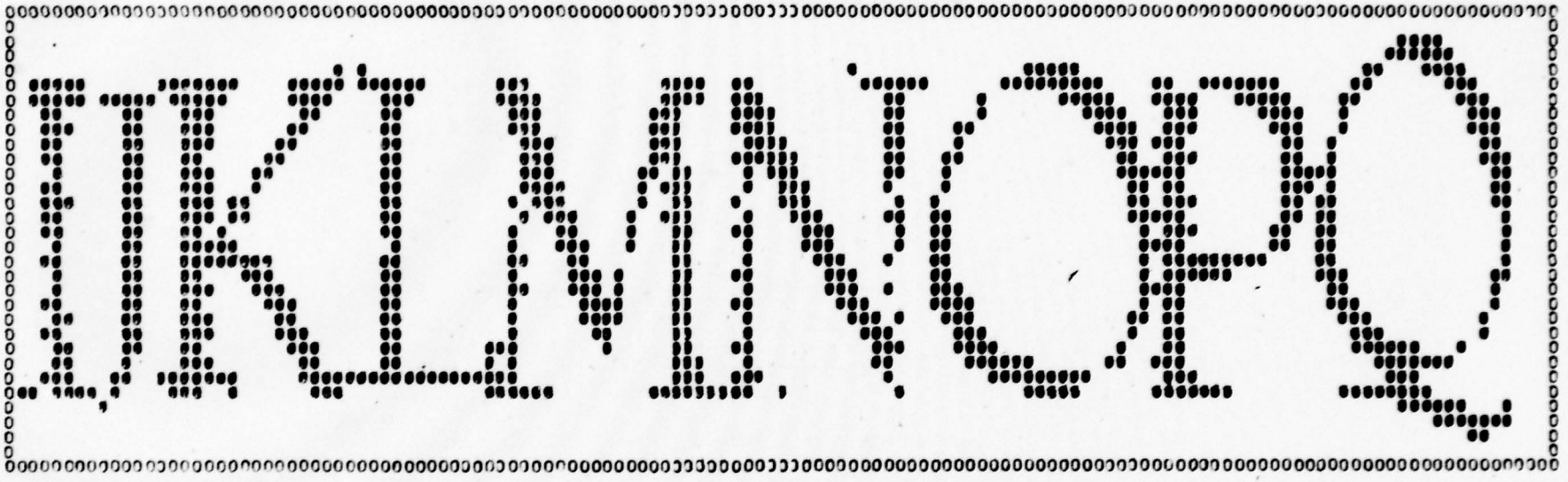
CHARACTERISTICS OF SENTENCE DATA: SENTENCE NUMBER 14.



SENTENCE # 14 COLUMN 1 TO 133
LETTERS APPEARING IN THE SENTENCE ARE:

COLUMNS	WIDTH	ALPHABET	LETTER
2- 17	16	13	1
18- 33	15	13	2
34- 47	14	13	3
48- 65	18	13	4
66- 77	12	13	5
78- 92	15	13	6
93-108	16	13	7
109-126	18	13	8

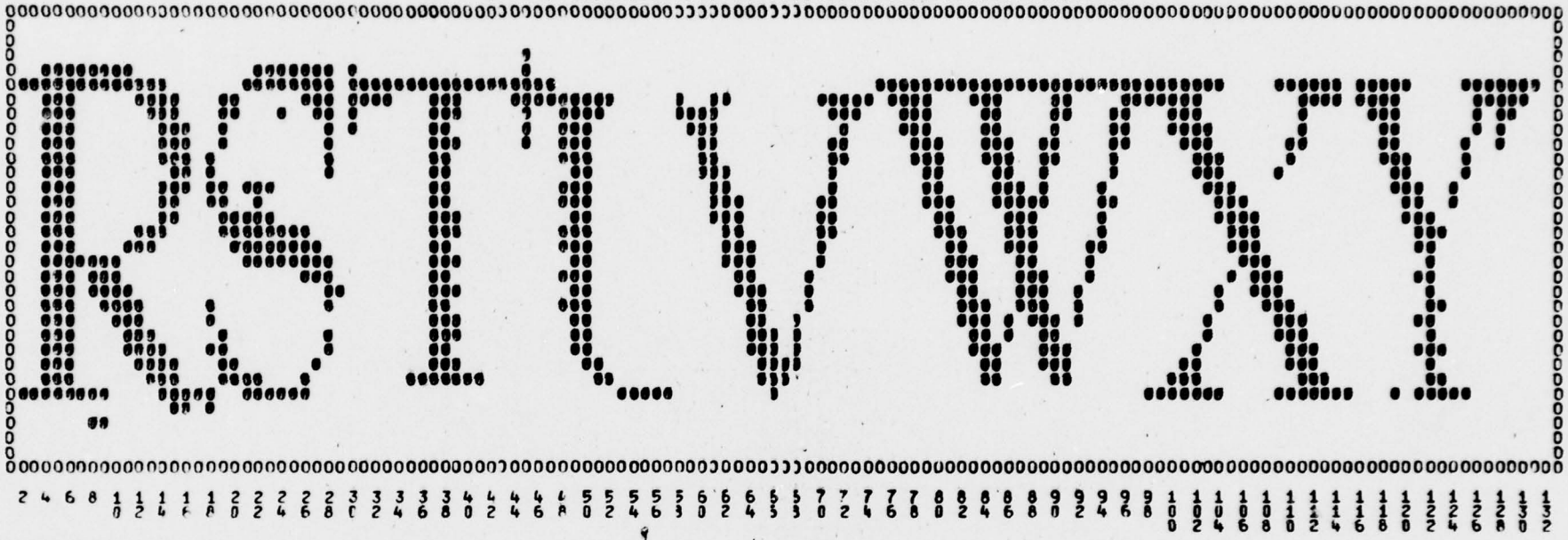
CHARACTERISTICS OF SENTENCE DATA: SENTENCE NUMBER 15.



SENTENCE # 15 COLUMN 1 TO 133
LETTERS APPEARING IN THE SENTENCE ARE:

COLUMNS	WIDTH	ALPHABET	LETTER
2- 8	7	13	9
9- 13	5	13	10
14- 30	17	13	11
31- 42	12	13	12
43- 62	20	13	13
63- 79	17	13	14
80- 98	19	13	15
99-112	14	13	16
113-129	17	13	17

CHARACTERISTICS OF SENTENCE DATA: SENTENCE NUMBER 15.



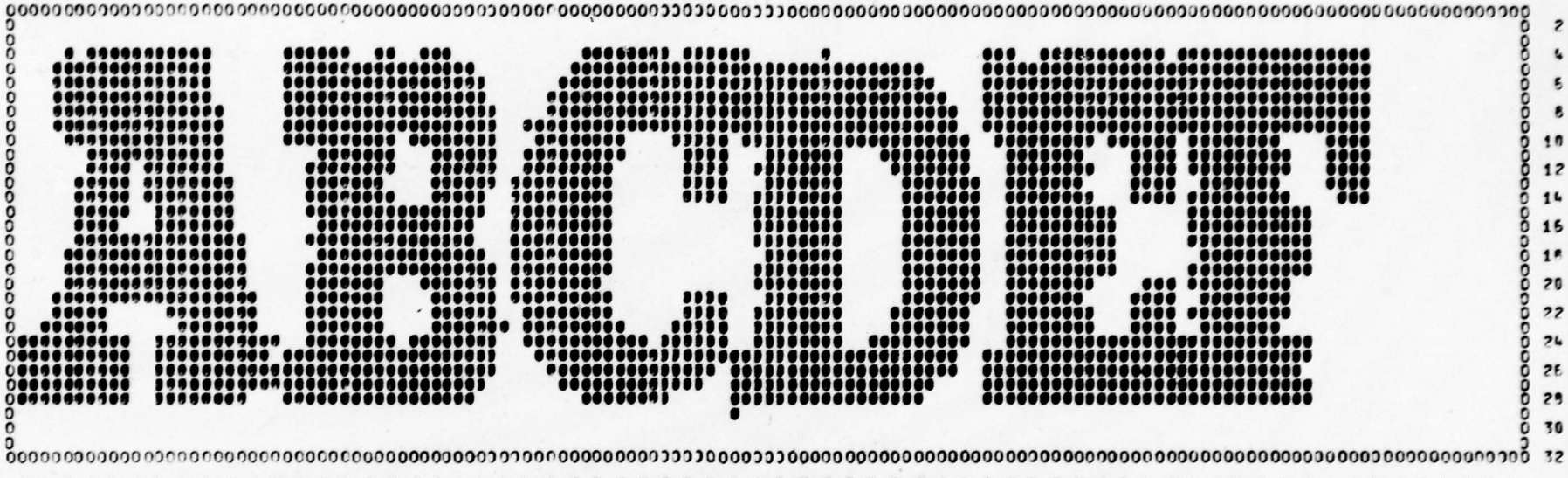
SENTENCE # 15 COLUMN 1 TO 133
LETTERS APPEARING IN THE SENTENCE ARE:

COLUMNS	WIDTH	ALPHABET	LETTER
2- 17	15	13	18
18- 29	12	13	19
30- 45	16	13	20
46- 57	12	13	21
58- 74	17	13	22
75- 97	23	13	23
98-115	18	13	24
115-131	15	13	25

SENTENCE # 17 COLUMN # TO #23
LETTERS APPEARING IN THE SENTENCE ARE:

COLUMNS WIDTH ALPHABET LETTER
2- 15 14 13 26

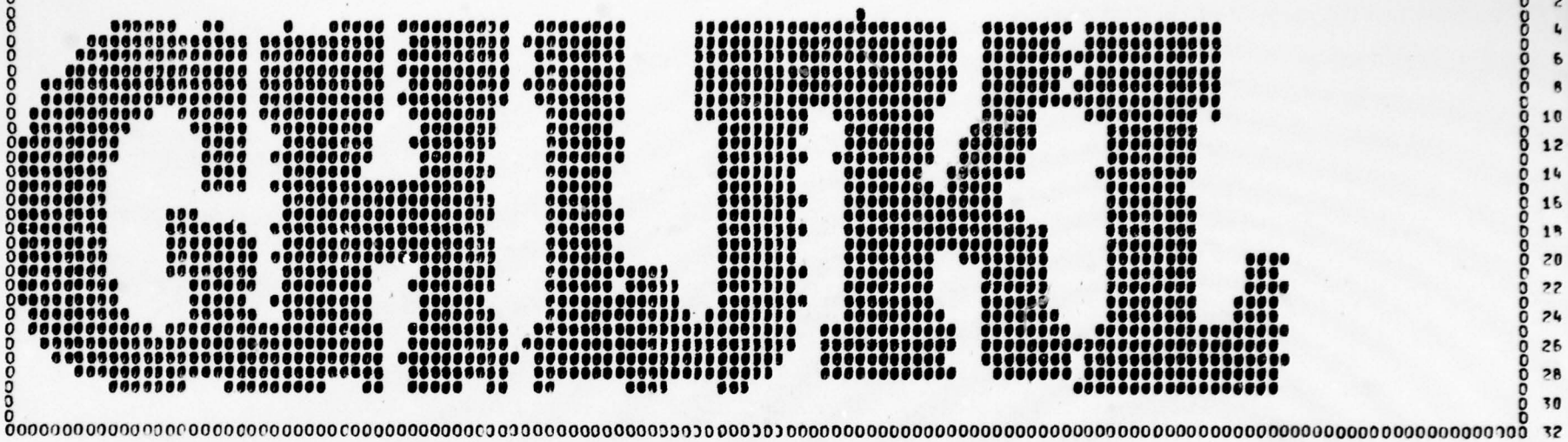
CHARACTERISTICS OF SENTENCE DATA! SENTENCE NUMBER 19.



SENTENCE # 19 COLUMN 1 TO 133
LETTERS APPEARING IN THE SENTENCE ARE:

COLUMNS	WIDTH	ALPHABET	LETTER
2- 24	23	14	1
25- 44	20	14	2
45- 63	19	14	3
64- 85	22	14	4
85-102	17	14	5
103-119	17	14	6

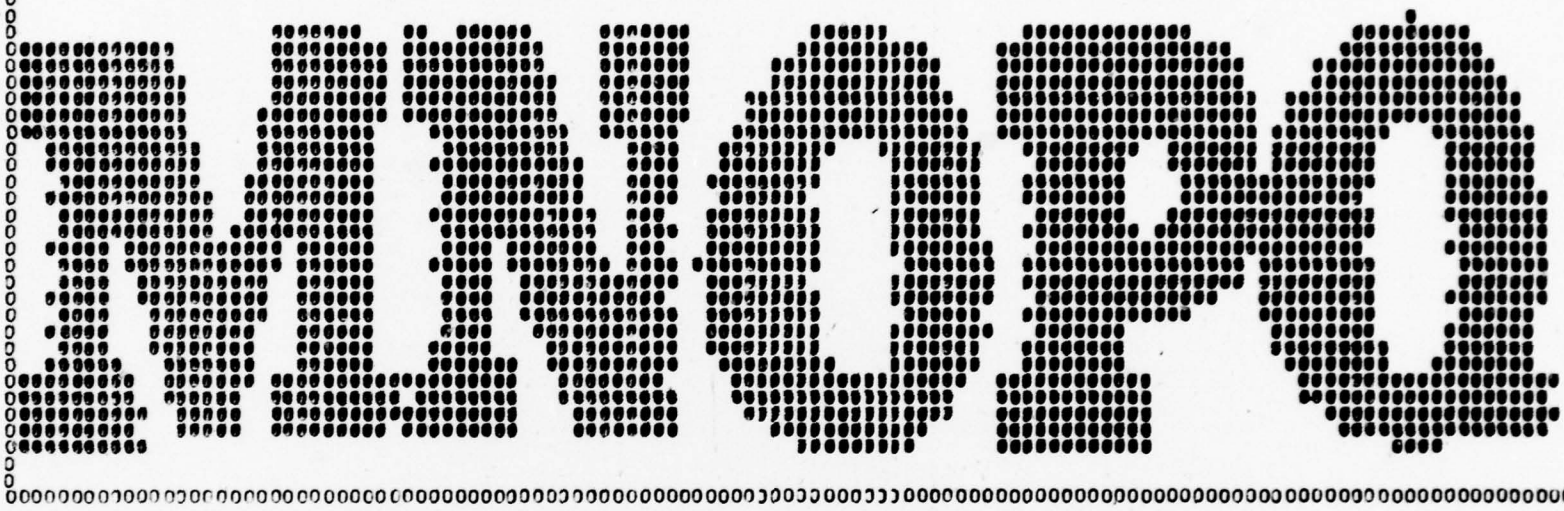
CHARACTERISTICS OF SENTENCE DATA! SENTENCE NUMBER 13.



SENTENCE # 13 COLUMN 1 TO 133
LETTERS APPEARING IN THE SENTENCE ARE!

COLUMNS	WIDTH	ALPHABET	LETTER
2- 22	21	14	7
23- 44	22	14	8
45- 54	10	14	9
55- 71	17	14	10
72- 93	22	14	11
94-117	13	14	12

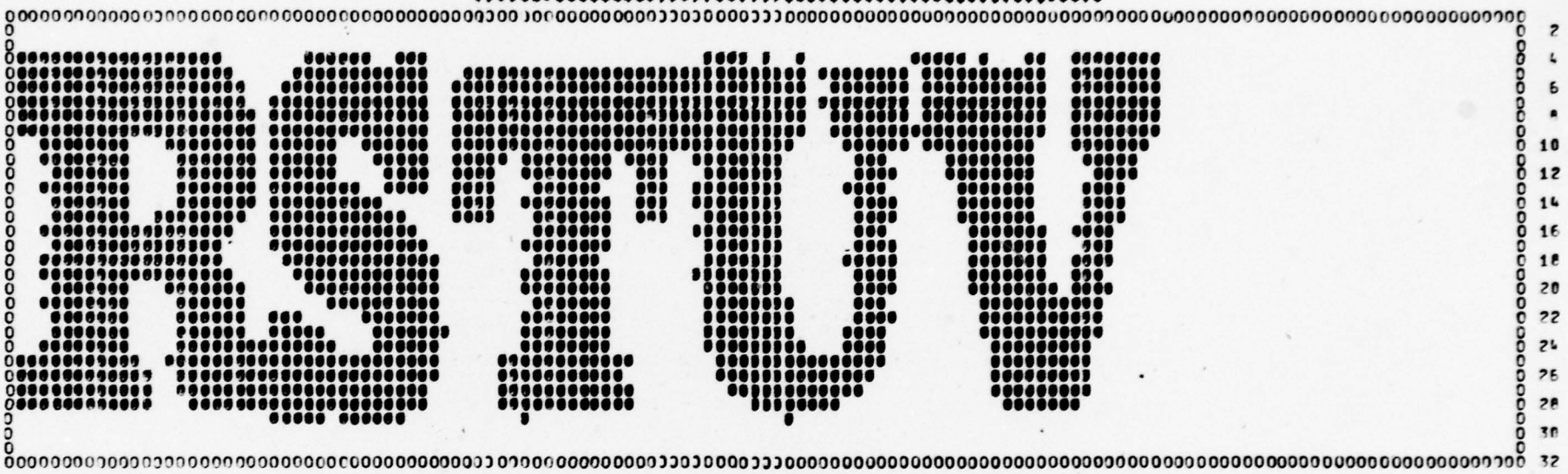
CHARACTERISTICS OF SENTENCE DATA: SENTENCE NUMBER 20.



SENTENCE # 20 COLUMN 1 TO 133
LETTERS APPEARING IN THE SENTENCE ARE:

COLUMNS	WIDTH	ALPHABET	LETTER
2- 30	29	14	13
31- 52	22	14	14
53- 75	23	14	15
75- 95	20	14	16
95-118	23	14	17

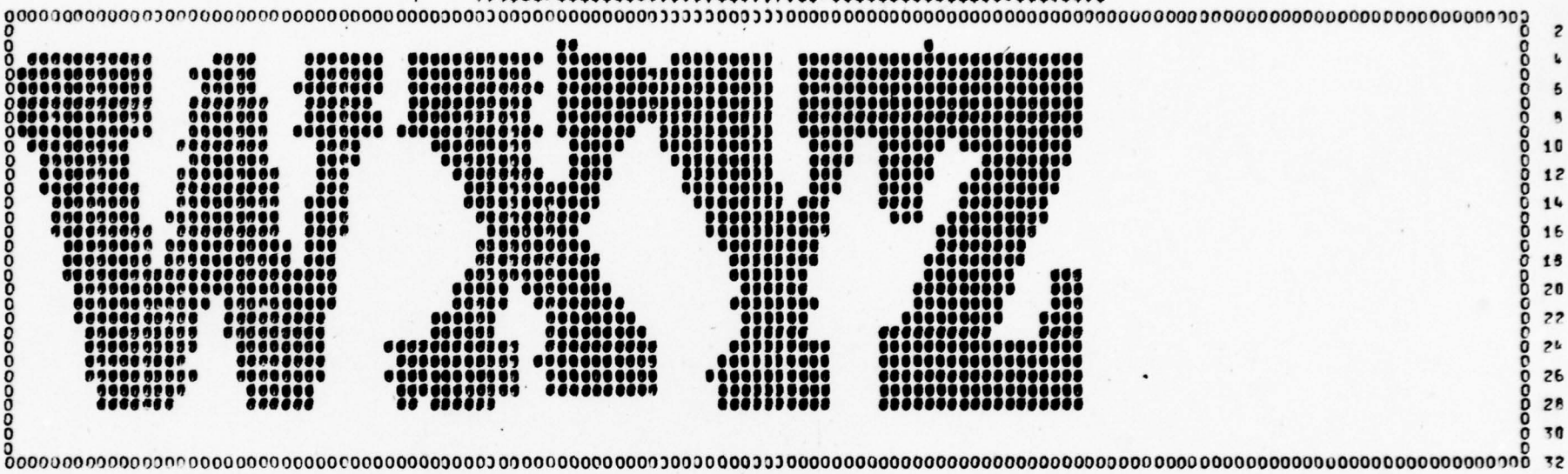
CHARACTERISTICS OF SENTENCE DATA! SENTENCE NUMBER 21.



SENTENCE # 21 COLUMN 1 TO 133
LETTERS APPEARING IN THE SENTENCE ARE!

COLUMNS	WIDTH	ALPHABET	LETTER
2- 23	22	14	18
24- 39	15	14	19
40- 58	19	14	20
59- 79	21	14	21
80-101	22	14	22

CHARACTERISTICS OF SENTENCE DATA! SENTENCE NUMBER 22



SENTENCE # 22 COLUMN 1 TO 133
LETTERS APPEARING IN THE SENTENCE ARE:

COLUMNS	WIDTH	ALPHABET LETTER	
2- 33	32	14	23
34- 57	24	14	24
58- 76	19	14	25
77- 94	18	14	26

Appendix C

MAKFLTR

```
      PROGRAM MAIN(INPUT,OUTPUT,TAPE3,TAPE4)
      INTEGER ALPHSTR,ALPHSTP,LTRSTRT,LTRSTOP,LTRSIZE,NSPACE,TAPEIN,
     %TAPEOUT  $ LOGICAL ADJUST,RANDIN  $  REAL WRATIO,HRATIO
      INTEGER ENORM,DC
      LOGICAL       LTR,VEXPAND,XFORM,IMAGE,FLTR,INVERSE,STORE
1     FORMAT(1HT)
      LTRSIZE=32  $  NSPACE=81  $  TAPEIN=3   $TAPEOUT=4
      ALPHSTR=1     $  ALPHSTP=150      $    LTRSTRT=1   $  LTRSTOP=26
      ADJUST=.F.    $  RANDIN=.T.  $  LTR=.F.  $  VEXPAND=.F. $XFORM=.F.
      IMAGE=.F.    $  FLTR=.F.  $  INVERSE=.F.  $    STORE=.T.
      ENORM=0  $   DC=0
      WRITE 1
      IF (RANDIN) CALL PREPLTR(TAPEIN)
      CALL PREPFL(TAPEOUT)
      CALL        FORTAPE(ALPHSTR,ALPHSTP,LTRSTRT,LTRSTOP,LTRSIZE,
     % NSPACE,ADJUST,WRATIO,HRATIO,TAPEIN,RANDIN,TAPEOUT,LTR,
     % VEXPAND,XFORM,IMAGE,FLTR,INVERSE,STORE,ENORM,DC)
      STOP
      END
      SUBROUTINE FORTAPE(ALPHSTR,ALPHSTP,LTRSTRT,LTRSTOP,LTRSIZE,
     % NSPACE,ADJUST,WRATIO,HRATIO,TAPEIN,RANDIN,TAPEOUT,LTR,
     % VEXPAND,XFORM,IMAGE,FLTR,INVERSE,STORE,ENORM,DC)
      INTEGER ALPHSTR,ALPHSTP,LTRSTRT,LTRSTOP,LTRSIZE,NSPACE,TAPEIN,
     %TAPEOUT  $ LOGICAL ADJUST,RANDIN  $  REAL WRATIO,HRATIO
      LOGICAL       LTR,VEXPAND,XFORM,IMAGE,FLTR,INVERSE,STORE
      INTEGER LETTER(32,32),NUMLTRS,RECNUM
      COMMON/LETTERS/LETTER,NUMLTRS,RECNUM
      INTEGER RECFLTR
      REAL FLTRLTR(180),PROTO(180)
      COMMON/PROFLTR/FLTRLTR,PROTO,RECFLTR
      COMMON/XFORMS/FORTLTR
      COMPLEX FORTLTR(64,64)
      INTEGER ALPHNUM,LTRNUM,        WIDTH,HEIGHT,NUMROWS,NUMCOLS
1     FORMAT(1H1)
3     FORMAT (1H ,"ALPHABET ",I3," COMPLETE.")
      WRITE 1
      NUMWRDS=NSPACE+4
      DO 110 ALPHNUM=ALPHSTR,ALPHSTP
      DO 100 LTRNUM=LTRSTRT,LTRSTOP
      CALL GETLTR(ALPHNUM,LTRNUM,TAPEIN,RANDIN)
      IF (LTR)            CALL PRNTLTR(ALPHNUM,LTRNUM,.F.,.T.,.F.,.T.)
      CALL XFORMIT(LTRSIZE,WIDTH,HEIGHT,ADJUST,WRATIO,HRATIO,NSPACE,
     % NUMROWS,NUMCOLS,ALPHNUM,LTRNUM,VEXPAND,XFORM,IMAGE,INVERSE)
      IF (XFORM)
     %CALL PRNTCPX(FORTLTR,NUMROWS,NUMCOLS,1,NUMROWS,1,NUMCOLS,
     %1,ALPHNUM  ,LTRNUM,0,.T.,.T.)
      CALL FILTER (FORTLTR,NUMROWS,NUMCOLS,FLTRLTR,NSPACE)
      IF (FLTR) CALL PRNTFLR(ALPHNUM,LTRNUM,FLTRLTR,NSPACE,3,.F.)
      IF (ENORM.EQ.1) CALL ENORMFL(FLTRLTR,FLTRLTR,NSPACE,DC)
      FLTRLTR(NSPACE+1)=WIDTH
      FLTRLTR(NSPACE+2)=HEIGHT
      FLTRLTR(NSPACE+3)=NUMROWS
      FLTRLTR(NSPACE+4)=NUMCOLS
      KEY=(ALPHNUM-1)*26+LTRNUM
      IF (STORE) CALL WRITMS(TAPEOUT,FLTRLTR,NUMWRDS,KEY)
100   CONTINUE
      IF (STORE) WRITE 3,(ALPHNUM)
110   CONTINUE
      IF (.NOT.RANDIN) REWIND TAPEIN
C     IF (STORE) REWIND TAPEOUT
      RETURN
```

```
      SUBROUTINE ARRGSPX(TEMP,NROWS,NCOLS,TAMP,NUMROWS,NUMCOLS)
      COMPLEX TEMP(NROWS,NCOLS),TAMP(NUMROWS,NUMCOLS)
      INTEGER NROWS,NCOLS,NUMROWS,NUMCOLS
      DO 100 NCOL=1,NUMCOLS
      DO 100 NROW=1,NUMROWS
      TAMP(NROW,NCOL)=TEMP(NROW,NCOL)
100   CONTINUE
      RETURN
      END
      SUBROUTINE ARRANGE(TEMP,NROWS,NCOLS,TAMP,NUMROWS,NUMCOLS)
      INTEGER TEMP(NROWS,NCOLS),TAMP(NUMROWS,NUMCOLS)
      INTEGER NROWS,NCOLS,NUMROWS,NUMCOLS
      DO 100 NCOL=1,NUMCOLS
      DO 100 NROW=1,NUMROWS
      TAMP(NROW,NCOL)=TEMP(NROW,NCOL)
100   CONTINUE
      RETURN
      END
      SUBROUTINE  ENORMFL (FLTRLTR,NORMLTR,NSPACE,DC)
      REAL FLTRLTR(NSPACE),NORMLTR(NSPACE)
      INTEGER NSPACE,DC
      INTEGER START,NVECTOR     $  REAL SUMSQRS
      START=2
      IF (DC.EQ.1) START=1
      SUMSQRS=0.0
      DO 100 NVECTOR=START,NSPACE
100   SUMSQRS=SUMSQRS+FLTRLTR(NVECTOR)**2
      SUMSQRS=SUMSQRS**.5
      IF (SUMSQRS.EQ.0.0) SUMSQRS=1.0
      DO 110 NVECTOR=START,NSPACE
110   NORMLTR(NVECTOR)=FLTRLTR(NVECTOR)/SUMSQRS
      IF (DC.EQ.0) NORMLTR(1)=1
      RETURN
      END
      SUBROUTINE FILTER(FORTLTR,NUMROWS,NUMCOLS,FLTRLTR,NSPACE)
      INTEGER NUMROWS,NUMCOLS,NSPACE
      COMPLEX FORTLTR(NUMROWS,NUMCOLS)
      REAL FLTRLTR(NSPACE)
      INTEGER NHRMNCS,NUMWRDS,RIGHT,LEFT,DOWN,NROW,NCOL
      NHRMNCS=NSPACE**.5
      RIGHT=NHRMNCS/2+1
      LEFT=NUMCOLS-RIGHT+2
      DOWN=RIGHT
      FLTRLTR(1)=REAL(FORTLTR(1,1))
      IF (NHRMNCS.EQ.1) RETURN
      NUMWRDS=0
      DO 100 NCOL=2,RIGHT
      NUMWRDS=NUMWRDS+2
      FLTRLTR(NUMWRDS)=REAL(FORTLTR(1,NCOL))
100   FLTRLTR(NUMWRDS +1 )=AIMAG(FORTLTR(1,NCOL))
      DO 220 NROW=2,DOWN
      DO 210 NCOL=LEFT,NUMCOLS
      NUMWRDS=NUMWRDS+2
      FLTRLTR(NUMWRDS  )=REAL(FORTLTR(NROW,NCOL))
      FLTRLTR(NUMWRDS+1)=AIMAG(FORTLTR(NROW,NCOL))
210   CONTINUE
      DO 200 NCOL=1,RIGHT
      NUMWRDS=NUMWRDS+2
      FLTRLTR(NUMWRDS  )=REAL(FORTLTR(NROW,NCOL))
      FLTRLTR(NUMWRDS+1)=AIMAG(FORTLTR(NROW,NCOL))
200   CONTINUE
220   CONTINUE
      RETURN
      END
      SUBROUTINE GETLTR (ALPHNUM,LTRNUM,TAPENUM,RANDOM)
      INTEGER ALPHNUM,LTRNUM,TAPENUM
      LOGICAL RANDOM
```

```
      LOGICAL RANDOM
      INTEGER LETTER(32,32),NUMLTRS,RECNUM
      COMMON/LETTERS/LETTER,NUMLTRS,RECNUM
      INTEGER RECRQD,RECSKP,MASKER,LTRSTRT,I,J,NROW,NCOL,ALPHPAC(832)
     % ,TEMP
      COMMON/LOADMS/ALPHPAC
      DATA RECNUM/1/
      IF (.NOT.RANDOM) GO TO 7
      LTRSTRT=(LTRNUM-1)*32+1
      KEY=(ALPHNUM-1)*26+LTRNUM
      CALL READMS(TAPENUM,ALPHPAC(LTRSTRT),32,KEY)
      GO TO 60
    7 CONTINUE
      RECRQD=ALPHNUM
      IF ((RECRQD-RECNUM).EQ.-1) GO TO 60
      RECSKP=IABS(RECRQD-RECNUM)
      IF (RECRQD-RECNUM) 10,30,20
   10 DO 15 I=1,RECSKP
   15 BACKSPACE TAPENUM
      GO TO 30
   20 DO 25 I=1,RECSKP
      BUFFER IN (TAPENUM,1) (ALPHPAC(1),ALPHPAC(832))
      CALL CHCKTPE(ALPHNUM,LTRNUM,TAPENUM)
   25 CONTINUE
   30 BUFFER IN (TAPENUM,1) (ALPHPAC(1),ALPHPAC(832))
      RECNUM=RECRQD+1
      CALL CHCKTPE(ALPHNUM,LTRNUM,TAPENUM)
   60 LTRSTRT=(LTRNUM-1)*32
      MASKER=MASK(1)
      DO 80 NROW=1,32
      TEMP=ALPHPAC(LTRSTRT+NROW)
      DO 70 NCOL=1,32
      IF (TEMP.AND.MASKER) 65,68
   65 LETTER(NROW,NCOL)=1
      GO TO 69
   68 LETTER(NROW,NCOL)=0
   69 TEMP=SHIFT(TEMP,1)
   70 CONTINUE
   80 CONTINUE
      RETURN
      END
      SUBROUTINE PREPLTR(LTRTAPE)
      INTEGER LTRTAPE
      INTEGER LINDEX(3901)
      CALL OPENMS(LTRTAPE,LINDEX,3901,0)
      RETURN
      END
      SUBROUTINE PREPFL(FLRTAP)
      INTEGER FLRTAP
      INTEGER MINDEX(3901)
      CALL OPENMS(FLRTAP,MINDEX,3901,0)
      RETURN
      END
      SUBROUTINE REPLACE(TRPLTR,FORTLTR,NUMROWS,NUMCOLS)
      INTEGER NUMROWS,NUMCOLS
      DIMENSION TRPLTR(32,32)
      COMPLEX FORTLTR(NUMROWS,NUMCOLS)
      DO 100 NROW=1,NUMROWS
      DO 100 NCOL=1,NUMCOLS
  100 FORTLTR(NROW,NCOL)=TRPLTR(NROW,NCOL)
      RETURN
      END
      SUBROUTINE SIZEIT(LTRSIZE,WIDTH,HEIGHT)
      INTEGER LTRSIZE,WIDTH,HEIGHT
      COMMON/LETTERS/LETTER,NUMLTRS,RECNUM
      INTEGER LETTER(32,32),NUMLTRS,RECNUM
      INTEGER NCOL,NROW,STRTCOL,TEMP,STRTROW
```

```
      DO 100 NCOL=1,32
      DO 100 NROW=1,32
      IF (LETTER(NROW,NCOL).EQ.1) GO TO 110
  100 CONTINUE
  110 STRTCOL=NCOL
      DO 120 NCOL=1,32
      TEMP=33-NCOL
      DO 120 NROW=1,32
      IF (LETTER(NROW, TEMP).EQ.1) GO TO 130
  120 CONTINUE
  130 WIDTH=TEMP-STRTCOL+1
      IF (WIDTH.GE.1)  GO TO 135
      WIDTH=HEIGHT=0
      RETURN
  135 CONTINUE
      DO 140 NROW=1,32
      DO 140 NCOL=1,LTRSIZE
      IF(LETTER(NROW,NCOL).EQ.1) GO TO 150
  140 CONTINUE
  150 STRTROW=NROW
      DO 160 NROW=1,LTRSIZE
      TEMP=LTRSIZE+1-NROW
      DO 160 NCOL=1,LTRSIZE
      IF (LETTER(TEMP,NCOL).EQ.1) GO TO 170
  160 CONTINUE
  170 HEIGHT=TEMP-STRTROW+1
      RETURN
      END
      SUBROUTINE CPXREAL(REALARY,CPXARY,NUMROWS,NUMCOLS)
      REAL REALARY(NUMROWS,NUMCOLS) $ INTEGER NUMROWS,NUMCOLS
      COMPLEX CPXARY(NUMROWS,NUMCOLS)
      DO 100 NROW=1,NUMROWS
      DO 100 NCOL=1,NUMCOLS
  100 REALARY(NROW,NCOL)=CPXARY(NROW,NCOL)
      RETURN
      END
      SUBROUTINE FRMTNUM(FMT,A,B,C)
      INTEGER FMT,A,B,C
      GO TO (10,20,30,40,50),FMT
    1 FORMAT(/" FOURIER TRANSFORM OF ALPHABET ",I3," LETTER " ,I2,/)
   10 WRITE 1,(A,B)
      RETURN
    2 FORMAT(/ " FOURIER TRANSFORM OF SENTENCE ",I2," STARTING AT COLUMN
     % ",I3,/)
   20 WRITE 2,(A,B)
      RETURN
    3 FORMAT (/," FOURIER XFORM OF ALPHABET ",I2," LETTER ",I2,
     % " FILTERED TO ",I3," SPACE",/)
   30 WRITE 3,(A,B,C)
      RETURN
    4 FORMAT (" ALPHABET ",I3," LETTER ",I2,/)
   40 WRITE 4,(A,B)
      RETURN
   50 CONTINUE
      RETURN
      END
      SUBROUTINE GLPRT(M,N,A,STORE,NEG,NFORM,SKIPAGE)
      DIMENSION DEN(20,7),A(M,N),STORE( N,7)
      DIMENSION FMT1(1),FMT2(2),FMT3(2)
      LOGICAL SKIPAGE
C     THE MAIN PROGRAM MUST SUPPLY FMT1,FMT2,& FMT3
C     FOR EXAMPLE IF M=32, USE THE FOLLOWING DATA CARDS IN MAIN PROGRAM.
C     DATA FMT1/20H(1H ,32I2)          /
C     DATA FMT2,FMT3/10H(1H ,32A1),10H(1H+,32A1)/
C     IF M IS LT 43, THE FOLLOWING CARD MAY BE USED.  E.G. M=32 IS USED.
C     DATA FMT1/20H(1H ,32I2)          /
```

```
C     NEG = 0 IS USED TO PRESENT THE MAX VALUE AS BLANK (WHITE);
C     WHILE NEG NOT = 0 IS TO PRESENT THE MAX VALUE AS BLACK.
C     THE CHOISE USUALLY DEPENDS UPON THE OPERATOR,S PREFERANCE
C     RECOMMEND NEG BE CHOSEN SUCH THAR THE BACKGROUND (MOST
C     OF THE FIELD) IS BLANK. IN OTHER WORDS, IF THE PICTURE IS
C     HIGH VALUES ON A LOW VALUE (BLACK) BACKGROUND, RECOMMEND
C     NEG NOT =0. THIS WILL CAUSE THE HIGH VALUES TO BE BLACK
C     AND THE BACKGROUND TO BE BLANK.
      INTEGER NEG,NFORM
      INTEGER STORE,DEN
      INTEGER DUMMY

      DATA FMT1/10H(1H ,64I2)/
      DATA FMT2/20H(1H ,"#",=(A1),"#") /
      DATA FMT3/20H(1H+,1X ,=(A1))     /
10    FORMAT(1H ,=("#"))
20    FORMAT(1H0)
      DATA(DEN(1,J),J=1,7)/7(1H ) /
      DATA(DEN(2,J),J=1,7)/1H-,6(1H )/
      DATA(DEN(3,J),J=1,7)/1H=,6(1H )/
      DATA(DEN(4,J),J=1,7)/1H+,6(1H )/
      DATA(DEN(5,J),J=1,7)/1H),6(1H )/
      DATA(DEN(6,J),J=1,7)/1H1,6(1H )/
      DATA(DEN(7,J),J=1,7)/1HZ,6(1H )/
      DATA(DEN(8,J),J=1,7)/1HX,6(1H )/
      DATA(DEN(9,J),J=1,7)/1HA,6(1H )/
      DATA(DEN(10,J),J=1,7)/1HM,6(1H )/
      DATA(DEN(11,J),J=1,7)/1HO,1H-,5(1H )/
      DATA(DEN(12,J),J=1,7)/1HO,1H=,5(1H )/
      DATA(DEN(13,J),J=1,7)/1HO,1H+,5(1H )/
      DATA(DEN(14,J),J=1,7)/1HO,1H),5(1H )/
      DATA(DEN(15,J),J=1,7)/1HO,1H+,1H.,4(1H )/
      DATA(DEN(16,J),J=1,7)/1HO,1H+,1H.,1H=,3(1H )/
      DATA(DEN(17,J),J=1,7)/1HO,1HX,1H',1H.,1H-,2(1H )/
      DATA(DEN(18,J),J=1,7)/1HO,1HX,1H.,1HH,1HC,2(1H )/
      DATA(DEN(19,J),J=1,7)/1HO,1HX,1H.,1HH,1HB,2(1H )/
      DATA(DEN(20,J),J=1,7)/1HO,1HX,1H.,1HH,1HB,1HV,1HA/
      DMAX=A(1,1)
      DMIN=A(1,1)
      DO 100 I=1,M
      DO 100 J=1,N
      DMAX=AMAX1(DMAX,A(I,J))
      DMIN=AMIN1(DMIN,A(I,J))
  100 CONTINUE
      WRITE 20
      IF (SKIPAGE) PRINT 1
      PRINT 4
      IF (NFORM .EQ. 0) GO TO 200
      WRITE 10,(N+2)
      DO 120 I=1,M
      L = 0
      DO 130 J=1,N
      IF(NEG.EQ.0) DUMMY = 20. - 19.*(A(I,J)-DMIN)/(DMAX-DMIN)
      IF(NEG.NE.0) DUMMY = 1. + 19.*(A(I,J)-DMIN)/(DMAX-DMIN)
      L = MAX0( DUMMY, L )
      IF(DUMMY.GT.20) DUMMY=20
      IF(DUMMY.LT.1) DUMMY=1
      DO 140 K=1,7
  140  STORE(J,K)=DEN(DUMMY,K)
  130  CONTINUE
      PRINT FMT2,N,(STORE(J,1),J=1,N)
      IF ( L .LE. 10 ) GO TO 120
      PRINT FMT3,N,(STORE(J,2),J=1,N)
      IF ( L .LE. 14 ) GO TO 120
      PRINT FMT3,N,(STORE(J,3),J=1,N)
      IF ( L .LE. 15 ) GO TO 120
```

```
      PRINT FMT3,N,(STORE(J,4),J=1,N)
      IF ( L .LE. 16 ) GO TO 120
      PRINT FMT3,N,(STORE(J,5),J=1,N)
      IF ( L .LE. 19 ) GO TO 120
      PRINT FMT3,N,(STORE(J,6),J=1,N)
      PRINT FMT3,N,(STORE(J,7),J=1,N)
  120 CONTINUE
      WRITE 10,(N+2)
      GO TO 225
  200 DO 220 I=1,M
      DO 210 J=1,N
      IF (NEG.EQ.0) DUMMY=99.-98.*(A(I,J)-DMIN)/(DMAX-DMIN)
      IF (NEG.NE.0) DUMMY= 1.+98.*(A(I,J)-DMIN)/(DMAX-DMIN)
      IF (DUMMY .GT. 99) DUMMY=99
      IF (DUMMY .LT. 1 ) DUMMY=1
  210 STORE(J,1)=DUMMY
      PRINT FMT1,(STORE(J,1),J=1,N)
  220 CONTINUE
  225 PRINT 3
    1 FORMAT(1H1)
    3 FORMAT(1H )
    4 FORMAT(1HT)
      RETURN
      END
      SUBROUTINE IVRTCPX(LETTER,NUMROWS,NUMCOLS)
      COMPLEX LETTER(NUMROWS,NUMCOLS)
      INTEGER NUMROWS,NUMCOLS
      REAL TEMP1,TEMP2
      DO 100 NROW=1,NUMROWS
      DO 100 NCOL=1,NUMCOLS
      J=NROW+NCOL
      IF (MOD(J,2).NE.1) GO TO 100
      TEMP1=REAL(LETTER(NROW,NCOL))*(-1.0)
      TEMP2=AIMAG(LETTER(NROW,NCOL))*(-1.0)
      LETTER(NROW,NCOL)=CMPLX(TEMP1,TEMP2)
  100 CONTINUE
      RETURN
      END
      SUBROUTINE PRNTCPX(ARRAY,ROWSIN,COLSIN,STRTROW,STOPROW,STRTCOL,
     % STOPCOL,FMT,A,B,C,SEQHRMC,SKIPAGE)
      INTEGER ROWSIN,COLSIN,STRTROW,STOPROW,STRTCOL,STOPCOL,FMT,A,B,C
      COMPLEX            ARRAY(ROWSIN,COLSIN)
      LOGICAL SEQHRMC,SKIPAGE
      INTEGER ROWSTRT,ROWSTOP,COLSTRT,COLSTOP,TEMP,NPAGE,NSPLITS,HALF,
     %CINDEX(50),RINDEX(50)
    1 FORMAT(1HT)
    5 FORMAT(1H1)
   10 FORMAT (/,5X, =(2X,I3,3X))
   20 FORMAT(1H0,I3,2X,=(F7.3,1X))
   30 FORMAT(1H ,5X, =(F7.3,1X))
   40 FORMAT(/)
   50 FORMAT(1H ,"ARRAY PRINTED IS FROM ROW ",I3," TO ",I3," COLUMN ",
     % I3," TO ",I3)
   70 FORMAT(1H ," INDEXING REQUESTED IS STRAIGHT SEQUENTIAL."//)
   80 FORMAT(1H ," INDEXING REQUESTED IS HARMONIC NUMBER."//)
      WRITE 1
      NUMCOLS=16
      IF (.NOT.SEQHRMC) GO TO 120
      DO 100 NCOL=STRTCOL,STOPCOL
  100 CINDEX(NCOL)=NCOL
      DO 110 NROW=STRTROW,STOPROW
  110 RINDEX(NROW)=NROW
      GO TO 170
  120 CONTINUE
      TEMP=STOPROW-STRTROW+1
      HALF=TEMP/2+STRTROW
```

```
      DO 130 NROW=STRTROW,HALF
      RINDEX(NROW)=TEMP
      TEMP=TEMP-2
130   CONTINUE
      HALF=HALF+1  $  TEMP=TEMP+2
      DO 140 NROW=HALF,STOPROW
      TEMP=TEMP+2
      RINDEX(NROW)=TEMP
140   CONTINUE
      TEMP=STOPCOL-STRTCOL+1
      HALF=TEMP/2+STRTCOL
      DO 150 NCOL=STRTCOL,HALF
      CINDEX(NCOL)=TEMP
      TEMP=TEMP-2
150   CONTINUE
      HALF=HALF+1  $  TEMP=TEMP+2
      DO 160 NCOL=HALF,STOPCOL
      TEMP=TEMP+2
      CINDEX(NCOL)=TEMP
160   CONTINUE
170   CONTINUE
      NSPLITS=(STOPCOL-STRTCOL)/NUMCOLS+1
      COLSTRT=STRTCOL
      DO 220 NPAGE=1,NSPLITS
      COLSTOP=COLSTRT+NUMCOLS-1
      IF (NSPLITS.EQ.NPAGE) COLSTOP=STOPCOL
      TEMP=COLSTOP-COLSTRT+1
      IF (SKIPAGE) WRITE 5
      WRITE 50,(STRTROW,STOPROW,COLSTRT,COLSTOP)
      IF (SEQHRMC)  180,190
180   WRITE 70
      GO TO 200
190   WRITE 80
200   CONTINUE
      CALL FRMTNUM(FMT,A,B,C)
      WRITE 10,(NUMCOLS,(CINDEX(NCOL),NCOL=COLSTRT,COLSTOP))
      DO 210 NROW=STRTROW,STOPROW
      WRITE 20,(RINDEX(NROW),TEMP,(REAL(ARRAY(NROW,NCOL)),NCOL=COLSTRT,
     % COLSTOP))
      WRITE 30,(TEMP,(AIMAG(ARRAY(NROW,NCOL)) ,NCOL=COLSTRT,COLSTOP))
210   CONTINUE
      COLSTRT=COLSTOP+1
      WRITE 40
220   CONTINUE
      RETURN
      END
      SUBROUTINE PRNTEXP(FORTLTR,NUMROWS,NUMCOLS,ALPHNUM,LTRNUM,
     % WRATIO,HRATIO,WIDTH,HEIGHT)
      COMPLEX FORTLTR(NUMROWS,NUMCOLS)
      INTEGER ALPHNUM,LTRNUM,WIDTH,HEIGHT,NUMCOLS,NUMROWS
      REAL WRATIO,HRATIO
1     FORMAT(1H0,"EXPANSION OF ALPHABET "I3" LETTER "I2/1H "WIDTH RATIO=
     %"F6.2"  HEIGHT RATIO="F6.2,/,1H "LETTER SIZE IS "I2" COLUMNS WIDE
     %BY " I2" ROWS HIGH"/1H ,"WINDOW SIZE IS "I2" COLUMNS WIDE BY "I2
     %" ROWS HIGH"///
     %,=(1H ,=(I1)/))
2     FORMAT(1HT)
      WRITE 2
                          WRITE 1,(ALPHNUM,LTRNUM,WRATIO,HRATIO,
     %WIDTH,HEIGHT,NUMCOLS,NUMROWS,NUMROWS,(NUMCOLS,(IFIX(REAL(FORTLTR
     %(NROW,NCOL)))     ,NCOL=1,NUMCOLS),NROW=1,NUMROWS))
      RETURN
      END
      SUBROUTINE PRNTFLR (ALPHNUM,LTRNUM,FLTRLTR,NSPACE,FMT,SKIPAGE)
      INTEGER ALPHNUM,LTRNUM,NSPACE,FMT  $  LOGICAL SKIPAGE
      REAL FLTRLTR(NSPACE)
```

```
      INTEGER NHRMNCS,NZEROES,DOWN,INDEX, NROW,START,STOP
      REAL EMPTY(7)
      INTEGER INDEX(13)
1     FORMAT(1HT)
5     FORMAT(1H1)
10    FORMAT (/,5X,13(4X,I2,4X))
20    FORMAT(1H-,I2,3X,13(F9.5,1X))
30    FORMAT(1H ,5X,13(F9.5,1X))
40    FORMAT(/)
50    FORMAT (/" NSPACE=",I3," WHICH IS INCORRECT (NOT AN ODD NUMBER) F
     %OR ALPHABET ",I3," LETTER ",I2,/)
      WRITE 1
      IF (MOD(NSPACE ,2).EQ.1)GO TO 90
      WRITE 50,(NSPACE,ALPHNUM,LTRNUM)
      RETURN
90    CONTINUE
      IF (SKIPAGE) WRITE 5
      DATA (INDEX(I),I=1,13)/13,11,9,7,5,3,1,3,5,7,9,11,13/
      DATA(EMPTY(I),I=1,7)/7*0.0/
      CALL FRMTNUM(FMT,ALPHNUM,LTRNUM,NSPACE)
      NHRMNCS=NSPACE**.5
      START=(13-NHRMNCS)/2+1
      STOP=START+NHRMNCS-1
      WRITE 10,(INDEX(I),I=START,STOP)
      DOWN=NHRMNCS/2+1
      NZEROES=NHRMNCS/2
      INDX=1
      WRITE 20,(INDX ,(EMPTY(I),I=1,NZEROES), FLTRLTR(1),
     % (FLTRLTR(NVECTOR),NVECTOR=2,NHRMNCS,2))
      NZEROES=NZEROES+1
      WRITE 30,(      (EMPTY(I),I=1,NZEROES),(FLTRLTR(NVECTOR),
     %  NVECTOR=3,NHRMNCS,2))
      START=NHRMNCS+1
      DO 100 NROW=2,DOWN
      INDX =INDX +2    $   STOP=START+NHRMNCS*2-1
      WRITE 20,(INDX,(FLTRLTR(NVECTOR),NVECTOR=START,STOP,2))
      START=START+1
      WRITE 30,(      FLTRLTR(NVECTOR),NVECTOR=START,STOP,2)
      START=STOP+1
100   CONTINUE
      WRITE 40
      RETURN
      END
      SUBROUTINE PRNTIMG(FORTLTR,NUMROWS,NUMCOLS,ALPHNUM,LTRNUM)
      COMPLEX FORTLTR(NUMROWS,NUMCOLS)  $  INTEGER NUMROWS,NUMCOLS
      INTEGER ALPHNUM,LTRNUM
      REAL STORE(64,7)
      REAL BRIGHT(64,64)
30    FORMAT(1H "FOURIER TRANSFORM OF ALPHABET "I3" LETTER "I2" CONVERTE
     %D TO BRIGHTNESS LEVELS"/,1H "ROWS="I3"  COLUMNS="I3)
      DO 92 NROW=1,NUMROWS
      DO 92 NCOL=1,NUMCOLS
92    BRIGHT(NROW,NCOL)=(REAL(FORTLTR(NROW,NCOL))**2+AIMAG(FORTLTR(NROW,
     % NCOL))**2)**.5
      CALL ARRANGE(BRIGHT,64,64,BRIGHT,NUMROWS,NUMCOLS)
      CALL GLPRT(NUMROWS,NUMCOLS,BRIGHT,STORE,1,1,.F.)
      WRITE 30,(ALPHNUM,LTRNUM,NUMROWS,NUMCOLS)
      RETURN
      END
      SUBROUTINE PRNTLTR(ALPHNUM,LTRNUM,DUMPIT,PRINTIT,ALPHSKP,SKIPAGE)
      INTEGER ALPHNUM,LTRNUM
      LOGICAL DUMPIT,PRINTIT,ALPHSKP,SKIPAGE
      INTEGER LETTER(32,32),NUMLTRS,RECNUM
      COMMON/LETTERS/LETTER,NUMLTRS,RECNUM
      INTEGER ADJCOL,NUMROW,NUMCOL,COLSIZE,NUMPAGE,OLDALPH,TITLE(72)
      INTEGER   KEEPER(32,100),KEEPER1(32,100)
```

```
      INTEGER*1 KEEPER(64,100),KEEPER1(64,100)
    1 FORMAT(1HT)
   98 FORMAT (1H+,     3(21X,I3,9X,I2,7X))
   99 FORMAT (/)
  100 FORMAT(/ ,1X,3(8X,3A1,1X,8A1,1X,3X,A1,1X,6A1 ,1X,2X,1X,3A1,3X))
  101 FORMAT(1H ,3(7X,33A1,2X))
  102 FORMAT(1H+,3(7X,32A1,1X,I2))
  103 FORMAT (/,1X,3(7X,16(1X,I1),3X))
  104 FORMAT (1H1)
  106 FORMAT (1X,3(7X,4(1X,A1),12(1X,I1),3X))
  107 FORMAT (1H+,3(7X,33A1,2X))
      DATA NUMLTRS,NUMPAGE/0,0/
      WRITE 1
      IF (NUMLTRS.EQ.0) OLDALPH=ALPHNUM
      IF (SKIPAGE) NUMPAGE=0
      IF ((NUMLTRS.EQ.0).AND.(NUMPAGE.EQ.0).OR .SKIPAGE) WRITE 10
      IF (PRINTIT) GO TO 105
      IF (DUMPIT.OR.(ALPHSKP.AND.(OLDALPH.NE.ALPHNUM))) 120,105
  105 NUMLTRS=NUMLTRS + 1
      ADJCOL=33*(NUMLTRS-1) +1
      COLSIZE=ADJCOL+31
      DO 117 NUMROW=1,32
      J=0
      DO 116 NUMCOL=ADJCOL,COLSIZE
      J=J+1
      IF (LETTER(NUMROW,J     ).EQ.0) 115,110
  110 KEEPER (NUMROW,NUMCOL)="#"
      KEEPER1(NUMROW,NUMCOL)="0"
      GO TO 116
  115 KEEPER(NUMROW,NUMCOL)=KEEPER1(NUMROW,NUMCOL)=" "
  116 CONTINUE
  117 CONTINUE
      TITLE(NUMLTRS*2+63)=LTRNUM
      TITLE(NUMLTRS*2+62)=ALPHNUM
      IF (PRINTIT) GO TO 120
      IF (NUMLTRS.LT.3) RETURN
  120 IF (NUMLTRS.EQ.0) RETURN
      COLSIZE=NUMLTRS*33
      DO 140 I=1,32
      DO 130 J=1,COLSIZE,33
  130 KEEPER1(I,J)="0"
      DO 135 J=32,COLSIZE,33
  135 KEEPER1(I,J)="0"
  140 CONTINUE
      DO 145 I=1,32,31
      DO 145 J=1,COLSIZE
  145 KEEPER1(I,J)="0"
      DO 165 I=2,32,2
      DO 165 J=33,COLSIZE,33
      KEEPER(I,J)=" "
  165 KEEPER1(I,J)=I
      DO 170 I=1,32,2
      DO 170 J=33,COLSIZE,33
  170 KEEPER(I,J)=KEEPER1(I,J)=" "
      DATA (TITLE(J),J=1,63)/3*("*","*","*","A","L","P","H","A","B",
     %"E","T",     ",","L","E","T","T","E","R",     "*","*","*")/
      DATA (KEEPER1(33,NUMCOL),NUMCOL=2,98,2)/3*(2,4,6,8,1,1,1,1,1,2,2,
     %2,2,2,3,3)/
      DATA (KEEPER1(34,NUMCOL),NUMCOL=2,98,2)/3*(" "," "," "," ",0,2,4,
     %6,8,0,2,4,6,8,0,2)/
      COLSIZE=NUMLTRS*21
      WRITE 100,(TITLE(NUMCOL),NUMCOL=1,COLSIZE)
      COLSIZE=NUMLTRS*2+63
      WRITE 98,(TITLE(I),I=64,COLSIZE)
      WRITE 99
      COLSIZE=NUMLTRS*33
      DO 180 I=1,32
```

```
      WRITE 101,(KEEPER (I,J),J=1,COLSIZE)
      IF (MOD(I,2).EQ.0) GO TO 175
      WRITE 107,(KEEPER1(I,J),J=1,COLSIZE)
      GO TO 180
  175 WRITE 102,(KEEPER1(I,J),J=1,COLSIZE)
  180 CONTINUE
      COLSIZE=32*NUMLTRS
      WRITE 103,(KEEPER1(33 ,NUMCOL),NUMCOL=2,COLSIZE,2)
      WRITE 106,(KEEPER1(34 ,NUMCOL),NUMCOL=2,COLSIZE,2)
      WRITE 99
      NUMLTRS=0
      NUMPAGE=NUMPAGE+1
      IF (ALPHSKP.AND.(OLDALPH.NE.ALPHNUM)) 190,195
  190 NUMPAGE=0
      OLDALPH=ALPHNUM
      WRITE 104
      GO TO 105
  195 IF ((NUMPAGE.EQ.2).OR.DUMPIT)  NUMPAGE=0
  200 RETURN
      END
      SUBROUTINE XFORMIT(LTRSIZE,WIDTH,HEIGHT,ADJUST,WRATIO,HRATIO,
     %NSPACE,NUMROWS,NUMCOLS,ALPHNUM,LTRNUM,VEXPAND,XFORM,IMAGE,INVERSE)
      INTEGER LTRSIZE,WIDTH,HEIGHT,NSPACE,NUMROWS,NUMCOLS,ALPHNUM,LTRNUM
      LOGICAL ADJUST    $ REAL WRATIO,HRATIO
      LOGICAL VEXPAND,XFORM,IMAGE,INVERSE
      INTEGER LETTER(32,32),NUMLTRS,RECNUM
      COMMON/LETTERS/LETTER,NUMLTRS,RECNUM
      INTEGER RECFLTR
      REAL FLTRLTR(180),PROTO(180)
      COMMON/PROFLTR/FLTRLTR,PROTO,RECFLTR
      COMMON/XFORMS/FORTLTR
      COMPLEX FORTLTR(64,64)
      COMPLEX WORKER(64)
      INTEGER POWERS(7)
      INTEGER NN(2),NUMCOLS,NUMROWS,NCOL,NROW,FLAG,TOP,BOTTOM,ITEMP,
     %              COL,ROW,      LEFT,RIGHT,IFORM,ISIGN
    2 FORMAT(1HT)
      DIMENSION TRPLTR(32,32)
      DATA POWERS/2,4,8,16,32,64,128/
      WRITE 2
      FLAG=0
   90 CONTINUE
      IF (ADJUST) GO TO 100
      NUMROWS=NUMCOLS=LTRSIZE
      WIDTH=32
      HEIGHT=22
      CALL TRPMLT(LETTER,TRPLTR)
      CALL REPLACE (TRPLTR,FORTLTR,NUMROWS,NUMCOLS)
      GO TO 235
  100 CONTINUE
      CALL SIZEIT(LTRSIZE,WIDTH,HEIGHT)
      NUMROWS=HEIGHT*HRATIO
      NUMCOLS=WIDTH*WRATIO
      NRINGS=NSPACE**.5
      IF(NUMROWS.LT.NRINGS) NUMROWS=NRINGS
      IF(NUMCOLS.LT.NRINGS) NUMCOLS=NRINGS
      IF (MOD(NUMROWS,2).EQ.1) NUMROWS=NUMROWS+1
      IF (MOD(NUMCOLS,2).EQ.1) NUMCOLS=NUMCOLS+1
      TOP =IABS(NUMROWS-LTRSIZE)/2
      IF (NUMROWS.LT.LTRSIZE) GO TO 150
      BOTTOM=TOP+1+LTRSIZE
      ITEMP=TOP
      DO 140 NCOL=1,LTRSIZE
      TOP=ITEMP
      DO 110 NROW=1,TOP
  110 FORTLTR(NROW,NCOL)=0
```

```
      DO 120 NROW=1,LTRSIZE
      TOP=TOP+1
120   FORTLTR(TOP,NCOL)=LETTER(NROW,NCOL)
      DO 130 NROW=BOTTOM,NUMROWS
130     FORTLTR(NROW,NCOL)=0
140   CONTINUE
      GO TO 170
150   CONTINUE
      TOP=TOP+1
      BOTTOM=TOP+NUMROWS-1
      DO 160 NCOL=1,LTRSIZE
      ROW=0
      DO 160 NROW=TOP,BOTTOM
      ROW=ROW+1
160   FORTLTR(ROW,NCOL)=LETTER(NROW,NCOL)
170   CONTINUE
      LEFT=IABS(NUMCOLS-LTRSIZE)/2
      IF(NUMCOLS.LT.LTRSIZE) GO TO 220
      RIGHT=LEFT+LTRSIZE+1
      DO 210 NROW=1,NUMROWS
      DO 180 NCOL=RIGHT,NUMCOLS
180   FORTLTR(NROW,NCOL)=0
      ITEMP=RIGHT
      DO 190 NCOL=1,LTRSIZE
      ITEMP=ITEMP-1
      COL=LTRSIZE-NCOL+1
190   FORTLTR(NROW,ITEMP)=FORTLTR(NROW,COL)
      DO 200 NCOL=1,LEFT
200   FORTLTR(NROW,NCOL)=0
210   CONTINUE
      GO TO 232
220   CONTINUE
      LEFT=LEFT+1
      RIGHT=LEFT+NUMCOLS-1
      DO 230 NROW=1,NUMROWS
      COL=0
      DO 230 NCOL=LEFT,RIGHT
      COL=COL+1
230   FORTLTR(NROW,COL)=FORTLTR(NROW,NCOL)
232   CONTINUE
      CALL ARNGCPX(FORTLTR,64,64,FORTLTR,NUMROWS,NUMCOLS)
235   CONTINUE
      IF (FLAG.GT.0) GO TO 245
      IF (VEXPAND) CALL PRNTEXP(FORTLTR,NUMROWS,NUMCOLS,ALPHNUM,LTRNUM,
     %WRATIO,HRATIO,WIDTH,HEIGHT)
      IF (XFORM.OR.IMAGE)
     % CALL IVRTCPX(FORTLTR,NUMROWS,NUMCOLS)
      NN(1)=NUMROWS $ NN(2)=NUMCOLS  $NDIM=2 $ ISIGN=-1 $ IFORM=0
      COL=ROW=0
      DO 240 ITEMP=1,7
      IF (NUMROWS.EQ.POWERS(ITEMP))  ROW=1
240   IF(NUMCOLS.EQ.POWERS(ITEMP))  COL=1
245   CONTINUE
      IF ( ROW.EQ.1.AND. COL.EQ.1) GO TO 250
      CALL FOURT (FORTLTR,NN,NDIM,ISIGN,IFORM,WORKER)
      GO TO 255
250   CALL FOURT (FORTLTR,NN,NDIM,ISIGN,IFORM,0.0)
C
C     THIS IS THE PLACE TO PUT IN AN INVERSE PRINT ROUTINE THAT
C     FILTERS THE FOURIER TRANSFORM FROM THE MIDDLE AND THEN CALL SLPRT
C
255   CONTINUE
      IF (FLAG.GT.0) RETURN
      IF (IMAGE) CALL PRNTIMG(FORTLTR,NUMROWS,NUMCOLS,ALPHNUM,LTRNUM)
      IF (XFORM)
     %CALL PRNTCPX(FORTLTR,NUMROWS,NUMCOLS,1,NUMROWS,1,NUMCOLS,
```

```
      %1,ALPHNUM  ,LTRNUM,0,.T.,.T.)
       FLAG=FLAG+1
       IF (XFORM.OP.IMAGE) GO TO 90
       RETURN
       END
       SUBROUTINE FOURT(DATA,NN,NDIM,ISIGN,IFORM,WORK)
C
C      THE COOLEY-TUKEY FAST FOURIER TRANSFORM IN USASI BASIC FORTRAN
C

C      TRANSFORM(K1,K2,...) = SUM(DATA(J1,J2,...)*EXP(ISIGN*2*PI*SQRT(-1)
C      *((J1-1)*(K1-1)/NN(1)+(J2-1)*(K2-1)/NN(2)+...))), SUMMED FOR ALL
C      J1, K1 BETWEEN 1 AND NN(1), J2, K2 BETWEEN 1 AND NN(2), ETC.
C      THERE IS NO LIMIT TO THE NUMBER OF SUBSCRIPTS.  DATA IS A
C      MULTIDIMENSIONAL COMPLEX ARRAY WHOSE REAL AND IMAGINARY
C      PARTS ARE ADJACENT IN STORAGE, SUCH AS FORTRAN IV PLACES THEM.
C      IF ALL IMAGINARY PARTS ARE ZERO (DATA ARE DISGUISED REAL), SET
C      IFORM TO ZERO TO CUT THE RUNNING TIME BY UP TO FORTY PERCENT.
C      OTHERWISE, IFORM = +1.  THE LENGTHS OF ALL DIMENSIONS ARE
C      STORED IN ARRAY NN, OF LENGTH NDIM.  THEY MAY BE ANY POSITIVE
C      INTEGERS, THO THE PROGRAM RUNS FASTER ON COMPOSITE INTEGERS, AND
C      ESPECIALLY FAST ON NUMBERS RICH IN FACTORS OF TWO.  ISIGN IS +1
C      OR -1.  IF A -1 TRANSFORM IS FOLLOWED BY A +1 ONE (OR A +1
C      BY A -1) THE ORIGINAL DATA REAPPEAR, MULTIPLIED BY NTOT (=NN(1)*
C      NN(2)*...).  TRANSFORM VALUES ARE ALWAYS COMPLEX, AND ARE RETURNED
C      IN ARRAY DATA, REPLACING THE INPUT.  IN ADDITION, IF ALL
C      DIMENSIONS ARE NOT POWERS OF TWO, ARRAY WORK MUST BE SUPPLIED,
C      COMPLEX OF LENGTH EQUAL TO THE LARGEST NON 2**K DIMENSION.
C      OTHERWISE, REPLACE WORK BY ZERO IN THE CALLING SEQUENCE.
C      NORMAL FORTRAN DATA ORDERING IS EXPECTED, FIRST SUBSCRIPT VARYING
C      FASTEST.  ALL SUBSCRIPTS BEGIN AT ONE.
C
C      RUNNING TIME IS MUCH SHORTER THAN THE NAIVE NTOT**2, BEING
C      GIVEN BY THE FOLLOWING FORMULA.  DECOMPOSE NTOT INTO
C      2**K2 * 3**K3 * 5**K5 * ....  LET SUM2 = 2*K2, SUMF = 3*K3 + 5*K5
C      + ... AND NF = K3 + K5 + ....  THE TIME TAKEN BY A MULTI-
C      DIMENSIONAL TRANSFORM ON THESE NTOT DATA IS T = T0 + NTOT*(T1+
C      T2*SUM2+T3*SUMF+T4*NF).  ON THE CDC 3300 (FLOATING POINT ADD TIME
C      OF SIX MICROSECONDS), T = 3000 + NTOT*(500+43*SUM2+68*SUMF+
C      320*NF) MICROSECONDS ON COMPLEX DATA.  IN ADDITION, THE
C      ACCURACY IS GREATLY IMPROVED, AS THE RMS RELATIVE ERROR IS

C      BOUNDED BY 3*2**(-B)*SUM(FACTOR(J)**1.5), WHERE B IS THE NUMBER
C      OF BITS IN THE FLOATING POINT FRACTION AND FACTOR(J) ARE THE
C      PRIME FACTORS OF NTOT.
C
C      PROGRAM BY NORMAN BRENNER FROM THE BASIC PROGRAM BY CHARLES
C      RADER.  RALPH ALTER SUGGESTED THE IDEA FOR THE DIGIT REVERSAL.
C      MIT LINCOLN LABORATORY, AUGUST 1967.  THIS IS THE FASTEST AND MOST
C      VERSATILE VERSION OF THE FFT KNOWN TO THE AUTHOR.  SHORTER PRO-
C      GRAMS FOUR1 AND FOUR2 RESTRICT DIMENSION LENGTHS TO POWERS OF TWO.
C      SEE-- IEEE AUDIO TRANSACTIONS (JUNE 1967), SPECIAL ISSUE ON FFT.
C
C      THE DISCRETE FOURIER TRANSFORM PLACES THREE RESTRICTIONS UPON THE
C      DATA.
C      1.  THE NUMBER OF INPUT DATA AND THE NUMBER OF TRANSFORM VALUES
C      MUST BE THE SAME.
C      2.  BOTH THE INPUT DATA AND THE TRANSFORM VALUES MUST REPRESENT
C      EQUISPACED POINTS IN THEIR RESPECTIVE DOMAINS OF TIME AND
C      FREQUENCY.  CALLING THESE SPACINGS DELTAT AND DELTAF, IT MUST BE
C      TRUE THAT DELTAF=2*PI/(NN(I)*DELTAT).  OF COURSE, DELTAT NEED NOT
C      BE THE SAME FOR EVERY DIMENSION.
C      3.  CONCEPTUALLY AT LEAST, THE INPUT DATA AND THE TRANSFORM OUTPUT
C      REPRESENT SINGLE CYCLES OF PERIODIC FUNCTIONS.
C
C      EXAMPLE 1.  THREE-DIMENSIONAL FORWARD FOURIER TRANSFORM OF A
```

```
C     EXAMPLE 1.  THREE-DIMENSIONAL FORWARD FOURIER TRANSFORM OF A
C     COMPLEX ARRAY DIMENSIONED 32 BY 25 BY 13 IN FORTRAN IV.
C     DIMENSION DATA(32,25,13),WORK(50),NN(3)
C     COMPLEX DATA
C     DATA NN/32,25,13/
C     DO 1 I=1,32
C     DO 1 J=1,25
C     DO 1 K=1,13
C
C  1  DATA(I,J,K)=COMPLEX VALUE
C     CALL FOURT(DATA,NN,3,-1,1,WORK)
C
C     EXAMPLE 2.  ONE-DIMENSIONAL FORWARD TRANSFORM OF A REAL ARRAY OF
C     LENGTH 64 IN FORTRAN II.
C     DIMENSION DATA(2,64)
C     DO 2 I=1,64
C     DATA(1,I)=REAL PART
C  2  DATA(2,I)=0.
C     CALL FOURT(DATA,64,1,-1,0,0)
C
      DIMENSION DATA(1),NN(1),IFACT(32),WORK(1)
      TWOPI=6.28318507
      WR=0.
      WI=0.
      WSTPI=0.
      WSTPR=0.
      IF(NDIM-1)920,1,1
1     NTOT=2
      DO 2 IDIM=1,NDIM
      IF(NN(IDIM))920,920,2
2     NTOT=NTOT*NN(IDIM)
C
C     MAIN LOOP FOR EACH DIMENSION
C
      NP1=2
      DO 910 IDIM=1,NDIM
      N=NN(IDIM)
      NP2=NP1*N
      IF(N-1)920,900,5
C
C     FACTOR N
C
5     M=N
      NTWO=NP1
      IF=1
      IDIV=2
10    IQUOT=M/IDIV
      IREM=M-IDIV*IQUOT
      IF(IQUOT-IDIV)50,11,11
11    IF(IREM)20,12,20

12    NTWO=NTWO+NTWO
      M=IQUOT
      GO TO 10
20    IDIV=3
30    IQUOT=M/IDIV
      IREM=M-IDIV*IQUOT
      IF(IQUOT-IDIV)60,31,31
31    IF(IREM)40,32,40
32    IFACT(IF)=IDIV
      IF=IF+1
      M=IQUOT
      GO TO 30
40    IDIV=IDIV+2
      GO TO 30
50    IF(IREM)60,51,60
51    NTWO=NTWO+NTWO
```

```
      NTWO=NTWO+NTWO
      GO TO 70
   60 IFACT(IF)=M
C
C     SEPARATE FOUR CASES--
C        1. COMPLEX TRANSFORM OR REAL TRANSFORM FOR THE 4TH, 5TH, ETC.
C           DIMENSIONS.
C        2. REAL TRANSFORM FOR THE 2ND OR 3RD DIMENSION.  METHOD--
C           TRANSFORM HALF THE DATA, SUPPLYING THE OTHER HALF BY CON-
C           JUGATE SYMMETRY.
C        3. REAL TRANSFORM FOR THE 1ST DIMENSION, N ODD.  METHOD--
C           TRANSFORM HALF THE DATA AT EACH STAGE, SUPPLYING THE OTHER
C           HALF BY CONJUGATE SYMMETRY.
C        4. REAL TRANSFORM FOR THE 1ST DIMENSION, N EVEN.  METHOD--
C           TRANSFORM A COMPLEX ARRAY OF LENGTH N/2 WHOSE REAL PARTS
C           ARE THE EVEN NUMBERED REAL VALUES AND WHOSE IMAGINARY PARTS
C           ARE THE ODD NUMBERED REAL VALUES.  SEPARATE AND SUPPLY
C           THE SECOND HALF BY CONJUGATE SYMMETRY.
C
   70 NON2=NP1*(NP2/NTWO)
      ICASE=1
      IF(IDIM-4)71,90,90
   71 IF(IFORM)72,72,90
   72 ICASE=2
      IF(IDIM-1)73,73,90
   73 ICASE=3
      IF(NTWO-NP1)90,90,74
   74 ICASE=4
      NTWO=NTWO/2
      N=N/2
      NP2=NP2/2
      NTOT=NTOT/2
      I=3
      DO 80 J=2,NTOT
      DATA(J)=DATA(I)
   80 I=I+2
   90 I1RNG=NP1
      IF(ICASE-2)100,95,100
   95 I1RNG=NP0*(1+NPREV/2)
C
C     SHUFFLE ON THE FACTORS OF TWO IN N.  AS THE SHUFFLING
C     CAN BE DONE BY SIMPLE INTERCHANGE, NO WORKING ARRAY IS NEEDED
C
  100 IF(NTWO-NP1)600,600,110
  110 NP2HF=NP2/2
      J=1
      DO 150 I2=1,NP2,NON2

      IF(J-I2)120,130,130
  120 I1MAX=I2+NON2-2
      DO 125 I1=I2,I1MAX,2
      DO 125 I3=I1,NTOT,NP2
      J3=J+I3-I2
      TEMPR=DATA(I3)
      TEMPI=DATA(I3+1)
      DATA(I3)=DATA(J3)
      DATA(I3+1)=DATA(J3+1)
      DATA(J3)=TEMPR
  125 DATA(J3+1)=TEMPI
  130 M=NP2HF
  140 IF(J-M)150,150,145
  145 J=J-M
      M=M/2
      IF(M-NON2)150,140,140
  150 J=J+M
C
```

```
C
C       MAIN LOOP FOR FACTORS OF TWO.  PERFORM FOURIER TRANSFORMS OF
C       LENGTH FOUR, WITH ONE OF LENGTH TWO IF NEEDED.  THE TWIDDLE FACTOR
C       W=EXP(ISIGN*2*PI*SQRT(-1)*M/(4*MMAX)).  CHECK FOR W=ISIGN*SQRT(-1)
C       AND REPEAT FOR W=ISIGN*SQRT(-1)*CONJUGATE(W).
C
        NON2T=NON2+NON2
        IPAR=NTWO/NP1
  310   IF(IPAR-2)350,330,320
  320   IPAR=IPAR/4
        GO TO 310
  330   DO 340 I1=1,I1RNG,2
        DO 340 J3=I1,NON2,NP1
        DO 340 K1=J3,NTOT,NON2T

        K2=K1+NON2
        TEMPR=DATA(K2)
        TEMPI=DATA(K2+1)
        DATA(K2)=DATA(K1)-TEMPR
        DATA(K2+1)=DATA(K1+1)-TEMPI
        DATA(K1)=DATA(K1)+TEMPR
  340   DATA(K1+1)=DATA(K1+1)+TEMPI
  350   MMAX=NON2
  360   IF(MMAX-NP2HF)370,600,600
  370   LMAX=MAX0(NON2T,MMAX/2)
        IF(MMAX-NON2)405,405,380
  380   THETA=-TWOPI*FLOAT(NON2)/FLOAT(4*MMAX)
        IF(ISIGN)400,390,390
  390   THETA=-THETA
  400   WR=COS(THETA)
        WI=SIN(THETA)
        WSTPR=-2.*WI*WI
        WSTPI=2.*WR*WI
  405   DO 570 L=NON2,LMAX,NON2T
        M=L
        IF(MMAX-NON2)420,420,410
  410   W2R=WR*WR-WI*WI
        W2I=2.*WR*WI
        W3R=W2R*WR-W2I*WI
        W3I=W2R*WI+W2I*WR
  420   DO 530 I1=1,I1RNG,2
        DO 530 J3=I1,NON2,NP1
        KMIN=J3+IPAR*M
        IF(MMAX-NON2)430,430,440
  430   KMIN=J3
  440   KDIF=IPAR*MMAX

  450   KSTEP=4*KDIF
        DO 520 K1=KMIN,NTOT,KSTEP
        K2=K1+KDIF
        K3=K2+KDIF
        K4=K3+KDIF
        IF(MMAX-NON2)460,460,480
  460   U1R=DATA(K1)+DATA(K2)
        U1I=DATA(K1+1)+DATA(K2+1)
        U2R=DATA(K3)+DATA(K4)
        U2I=DATA(K3+1)+DATA(K4+1)
        U3R=DATA(K1)-DATA(K2)
        U3I=DATA(K1+1)-DATA(K2+1)
        IF(ISIGN)470,475,475
  470   U4R=DATA(K3+1)-DATA(K4+1)
        U4I=DATA(K4)-DATA(K3)
        GO TO 510
  475   U4R=DATA(K4+1)-DATA(K3+1)
        U4I=DATA(K3)-DATA(K4)
        GO TO 510
  480   T2R=W2R*DATA(K2)-W2I*DATA(K2+1)
```

```
      T2I=W2R*DATA(K2+1)+W2I*DATA(K2)
      T3R=WR*DATA(K3)-WI*DATA(K3+1)
      T3I=WR*DATA(K3+1)+WI*DATA(K3)
      T4R=W3R*DATA(K4)-W3I*DATA(K4+1)
      T4I=W3R*DATA(K4+1)+W3I*DATA(K4)
      U1R=DATA(K1)+T2R
      U1I=DATA(K1+1)+T2I
      U2R=T3R+T4R
      U2I=T3I+T4I
      U3R=DATA(K1)-T2R
      U3I=DATA(K1+1)-T2I

      IF(ISIGN)490,500,500
  490 U4R=T3I-T4I
      U4I=T4R-T3R
      GO TO 510
  500 U4R=T4I-T3I
      U4I=T3R-T4R
  510 DATA(K1)=U1R+U2R
      DATA(K1+1)=U1I+U2I
      DATA(K2)=U3R+U4R
      DATA(K2+1)=U3I+U4I
      DATA(K3)=U1R-U2R
      DATA(K3+1)=U1I-U2I
      DATA(K4)=U3R-U4R
  520 DATA(K4+1)=U3I-U4I
      KMIN=4*(KMIN-J3)+J3
      KDIF=KSTEP
      IF(KDIF-NP2)450,530,530
  530 CONTINUE
      M=MMAX-M
      IF(ISIGN)540,550,550
  540 TEMPR=WR
      WR=-WI
      WI=-TEMPR
      GO TO 550
  550 TEMPR=WR
      WR=WI
      WI=TEMPR
  560 IF(M-LMAX)565,565,410
  565 TEMPR=WR
      WR=WR*WSTPR-WI*WSTPI+WR
  570 WI=WI*WSTPR+TEMPR*WSTPI+WI

      IPAR=3-IPAR
      MMAX=MMAX+MMAX
      GO TO 350
C
C     MAIN LOOP FOR FACTORS NOT EQUAL TO TWO.  APPLY THE TWIDDLE FACTOR
C     W=EXP(ISIGN*2*PI*SQRT(-1)*(J2-1)*(J1-J2)/(NP2*IFP1)), THEN
C     PERFORM A FOURIER TRANSFORM OF LENGTH IFACT(IF), MAKING USE OF
C     CONJUGATE SYMMETRIES.
C
  600 IF(NTWO-NP2)605,700,700
  605 IFP1=NON2
      IF=1
      NP1HF=NP1/2
  610 IFP2=IFP1/IFACT(IF)
      J1RNG=NP2
      IF(ICASE-3)612,611,612
  611 J1RNG=(NP2+IFP1)/2
      J2STP=NP2/IFACT(IF)
      J1RG2=(J2STP+IFP2)/2
  612 J2MIN=1+IFP2
      IF(IFP1-NP2)615,640,640
```

```
      DO 699 J2=J2MIN,IFP1,IFP2
      THETA=-TWOPI*FLOAT(J2-1)/FLOAT(NP2)
      IF(ISIGN)625,620,620
  620 THETA=-THETA
  625 SINTH=SIN(THETA/2.)
      WSTPR=-2.*SINTH*SINTH
      WSTPI=SIN(THETA)
      WR=WSTPR+1.
      WI=WSTPI
      J1MIN=J2+IFP1
      DO 635 J1=J1MIN,J1RNG,IFP1
      I1MAX=J1+I1RNG-2
      DO 630 I1=J1,I1MAX,2
      DO 630 I3=I1,NTOT,NP2
      J3MAX=I3+IFP2-NP1
      DO 630 J3=I3,J3MAX,NP1
      TEMPR=DATA(J3)
      DATA(J3)=DATA(J3)*WR-DATA(J3+1)*WI
  630 DATA(J3+1)=TEMPR*WI+DATA(J3+1)*WR
      TEMPR=WR
      WR=WR*WSTPR-WI*WSTPI+WR
  635 WI=TEMPR*WSTPI+WI*WSTPR+WI
  640 THETA=-TWOPI/FLOAT(IFACT(IF))
      IF(ISIGN)650,645,645
  645 THETA=-THETA
  650 SINTH=SIN(THETA/2.)
      WSTPR=-2.*SINTH*SINTH
      WSTPI=SIN(THETA)
      KSTEP=2*N/IFACT(IF)
      KRANG=KSTEP*(IFACT(IF)/2)+1
      DO 698 I1=1,I1RNG,2
      DO 698 I3=I1,NTOT,NP2
      DO 690 KMIN=1,KRANG,KSTEP
      J1MAX=I3+J1RNG-IFP1
      DO 680 J1=I3,J1MAX,IFP1
      J3MAX=J1+IFP2-NP1
      DO 680 J3=J1,J3MAX,NP1
      J2MAX=J3+IFP1-IFP2
      K=KMIN+(J3-J1+(J1-I3)/IFACT(IF))/NP1HF
      IF(KMIN-1)655,655,665
  655 SUMR=0.

      SUMI=0.
      DO 660 J2=J3,J2MAX,IFP2
      SUMR=SUMR+DATA(J2)
  660 SUMI=SUMI+DATA(J2+1)
      WORK(K)=SUMR
      WORK(K+1)=SUMI
      GO TO 680
  665 KCONJ=K+2*(N-KMIN+1)
      J2=J2MAX
      SUMR=DATA(J2)
      SUMI=DATA(J2+1)
      OLDSR=0.
      OLDSI=0.
      J2=J2-IFP2
  670 TEMPR=SUMR
      TEMPI=SUMI
      SUMR=TWOWR*SUMR-OLDSR+DATA(J2)
      SUMI=TWOWR*SUMI-OLDSI+DATA(J2+1)
      OLDSR=TEMPR
      OLDSI=TEMPI
      J2=J2-IFP2
      IF(J2-J3)675,675,670
  675 TEMPR=WR*SUMR-OLDSR+DATA(J2)
      TEMPI=WI*SUMI
      WORK(K)=TEMPR-TEMPI
```

```
      WORK(KCONJ)=TEMPR+TEMPI
      TEMPR=WR*SUMI-OLDSI+DATA(J2+1)
      TEMPI=WI*SUMR
      WORK(K+1)=TEMPR+TEMPI
      WORK(KCONJ+1)=TEMPR-TEMPI
  680 CONTINUE

      IF(KMIN-1)685,685,686
  685 WR=WSTPR+1.
      WI=WSTPI
      GO TO 690
  686 TEMPR=WR
      WR=WR*WSTPR-WI*WSTPI+WR
      WI=TEMPR*WSTPI+WI*WSTPR+WI
  690 TWOWR=WR+WR
      IF(ICASE-3)692,691,692
  691 IF(IFP1-NP2)695,692,692
  692 K=1
      I2MAX=I3+NP2-NP1
      DO 693 I2=I3,I2MAX,NP1
      DATA(I2)=WORK(K)
      DATA(I2+1)=WORK(K+1)
  693 K=K+2
      GO TO 698
C
C     COMPLETE A REAL TRANSFORM IN THE 1ST DIMENSION, N ODD, BY CON-
C     JUGATE SYMMETRIES AT EACH STAGE.
C
  695 J3MAX=I3+IFP2-NP1
      DO 697 J3=I3,J3MAX,NP1
      J2MAX=J3+NP2-J2STP
      DO 697 J2=J3,J2MAX,J2STP
      J1MAX=J2+J1RG2-IFP2
      J1CNJ=J3+J2MAX+J2STP-J2
      DO 697 J1=J2,J1MAX,IFP2
      K=1+J1-I3
      DATA(J1)=WORK(K)
      DATA(J1+1)=WORK(K+1)

      IF(J1-J2)697,697,696
  696 DATA(J1CNJ)=WORK(K)
      DATA(J1CNJ+1)=-WORK(K+1)
  697 J1CNJ=J1CNJ-IFP2
  698 CONTINUE
      IF=IF+1
      IFP1=IFP2
      IF(IFP1-NP1)700,700,610
C
C     COMPLETE A REAL TRANSFORM IN THE 1ST DIMENSION, N EVEN, BY CON-
C     JUGATE SYMMETRIES.
C
  700 GO TO (900,800,900,701),ICASE
  701 NHALF=N
      N=N+N
      THETA=-TWOPI/FLOAT(N)
      IF(ISIGN)703,702,702
  702 THETA=-THETA
  703 SINTH=SIN(THETA/2.)
      WSTPR=-2.*SINTH*SINTH
      WSTPI=SIN(THETA)
      WR=WSTPR+1.
      WI=WSTPI
      IMIN=3
      JMIN=2*NHALF-1
      GO TO 725
  710 J=JMIN
```

```
      DO 720 I=IMIN,NTOT,NP2
      SUMR=(DATA(I)+DATA(J))/2.
      SUMI=(DATA(I+1)+DATA(J+1))/2.
      DIFR=(DATA(I)-DATA(J))/2.

      DIFI=(DATA(I+1)-DATA(J+1))/2.
      TEMPR=WR*SUMI+WI*DIFR
      TEMPI=WI*SUMI-WR*DIFR
      DATA(I)=SUMR+TEMPR
      DATA(I+1)=DIFI+TEMPI
      DATA(J)=SUMR-TEMPR
      DATA(J+1)=-DIFI+TEMPI
  720 J=J+NP2
      IMIN=IMIN+2
      JMIN=JMIN-2
      TEMPR=WR
      WR=WR*WSTPR-WI*WSTPI+WR
      WI=TEMPR*WSTPI+WI*WSTPR+WI
  725 IF(IMIN-JMIN)710,730,740
  730 IF(ISIGN)731,740,740
  731 DO 735 I=IMIN,NTOT,NP2
  735 DATA(I+1)=-DATA(I+1)
  740 NP2=NP2+NP2
      NTOT=NTOT+NTOT
      J=NTOT+1
      IMAX=NTOT/2+1
  745 IMIN=IMAX-2*NHALF
      I=IMIN
      GO TO 755
  750 DATA(J)=DATA(I)
      DATA(J+1)=-DATA(I+1)
  755 I=I+2
      J=J-2
      IF(I-IMAX)750,760,760
  760 DATA(J)=DATA(IMIN)-DATA(IMIN+1)
      DATA(J+1)=0.

      IF(I-J)770,780,780
  765 DATA(J)=DATA(I)
      DATA(J+1)=DATA(I+1)
  770 I=I-2
      J=J-2
      IF(I-IMIN)775,775,765
  775 DATA(J)=DATA(IMIN)+DATA(IMIN+1)
      DATA(J+1)=0.
      IMAX=IMIN
      GO TO 745
  780 DATA(1)=DATA(1)+DATA(2)
      DATA(2)=0.
      GO TO 900
C
C     COMPLETE A REAL TRANSFORM FOR THE 2ND OR 3RD DIMENSION BY
C     CONJUGATE SYMMETRIES.
C
  800 IF(I1RNG-NP1)805,900,900
  805 DO 860 I3=1,NTOT,NP2
      I2MAX=I3+NP2-NP1
      DO 860 I2=I3,I2MAX,NP1
      IMIN=I2+I1RNG
      IMAX=I2+NP1-2
      JMAX=2*I3+NP1-IMIN
      IF(I2-I3)820,820,810
  810 JMAX=JMAX+NP2
  820 IF(IDIM-2)850,850,830
  830 J=JMAX+NP0
      DO 840 I=IMIN,IMAX,2
```

```
      DATA(I)=DATA(J)
      DATA(I+1)=-DATA(J+1)

  840 J=J-2
  850 J=JMAX
      DO 860 I=IMIN,IMAX,NP0
      DATA(I)=DATA(J)
      DATA(I+1)=-DATA(J+1)
  860 J=J-NP0
C
C     END OF LOOP ON EACH DIMENSION
C
  900 NP0=NP1
      NP1=NP2
  910 NPREV=N
  920 RETURN
      END
      SUBROUTINE TRPMLT(LETTER,TRPLTR)
      INTEGER LETTER(32,32)
      DIMENSION TRPLTR(32,32),TRPWDW(32)
    1 FORMAT(1H1)
    2 FORMAT(1H ,24("*"),/1H ,"* TRAPEZOID PARAMETERS *",/1H ,24("*"))
    3 FORMAT(1H ,/," LENGTH OF TRAPEZOID PLATEAU = ",I2)
    4 FORMAT(1H ,/," VARIANCE IS = ",F5.2)
    5 FORMAT(1H ,/," TRAPEZOID WINDOW ARRAY IS: ")
    6 FORMAT("       ",10F6.3)
    7 FORMAT(1H ,/," THIS IS A NORMALIZED WINDOW ")
      IF (PSKIP.EQ."OK") GO TO 45
CDATA NORMALIZED WINDOW SET IPLEN=0 AND TRPHT=VARIANCE
      IPLEN = 0     $     TRPHT = 5.0
C
C THIS PART BUILDS THE TRAPEZOID
C
      INCLN=(32-IPLEN)/2
      IF (IPLEN.NE.0) GO TO 9
      DO 8 I=1,16
    8 TRPWDW(17-I)=TRPWDW(16+I)=EXP(-((I-1)**2)/(2.*((TRPHT)**2)))
      WRITE 1
      WRITE 7
      GO TO 44
    9 IF (INCLN.EQ.0) GO TO 35
      DO 10 I=1,INCLN
      TRPWDW(I)=TRPWDW(33-I)=(I/(INCLN+1.))*TRPHT
   10 CONTINUE
      DO 20 I=1,IPLEN
      TRPWDW(I+INCLN)=1.
   20 CONTINUE
      IF((32-(2*INCLN)).EQ.IPLEN) GO TO 43
      IODD=INCLN+1
      DO 30 I=1,IODD
      TRPWDW(33-I)=(I/(INCLN+2.))*TRPHT
   30 CONTINUE
      GO TO 43
   35 DO 40 I=1,16
   40 TRPWDW(I)=TRPWDW(33-I)=1.
   43 CONTINUE
      WRITE 1
      WRITE 2
      WRITE 3,IPLEN
   44 WRITE 4,TRPHT
      WRITE 5
      WRITE 6,TRPWDW
C
C THIS PART DOES THE MULTIPLICATION
C
```

```
      DO 60 I=1,32
      DO 50 J=1,32
   50 TRPLTR(J,I)=TRPWDW(I)*(FLOAT(LETTER(J,I)))
   60 CONTINUE
      PSKIP="OK"
      RETURN
      END
```

Appendix D

PROTOBLD

```
      PROGRAM MAIN(INPUT,OUTPUT,TAPE4,TAPE17)
C     PROGRAM MAIN(INPUT=402B,OUTPUT=1002B,TAPE4=300B,TAPE10=300B)
1     FORMAT (1HT)
      INTEGER ALPHSTR,ALPHSTP,EXCPTNS(20),NEXCPTS ,TOTALPH,LTRSTRT,
     % LTRSTOP,NUMSUB,RANGE(2,10),NSPACE(10),TAPEIN,TAPEOUT,ENORM,DC
     %,LSPACE
      LOGICAL RANDIN,NEWFILE,ADD,CHANGE,DELETE
      WRITE 1
      ALPHSTR=1  $ ALPHSTP=150 $ LTRSTRT=1  $ LTRSTOP=26 $ TAPEIN=4
      RANDIN=.T. $ NEWFILE=.T.$ ADD=.F. $ CHANGE=.F. $ DELETE=.F.
      ENORM=1 $ DC=0 $ TAPEOUT=17 $ NUMSUB=1  $  NEXCPTS=0$  LSPACE=81
      XXXX=99 $ ENORM=1$ NUMSUB=1$ LSPACE=81$ NEXCPTS=6
      DATA EXCPTNS/24,62,82,101,106/,XXXX/1./
      DATA NSPACE/10*81/
      DATA RANGE/1,32/
      CALL PREPFLR(4)
      CALL PREPRTO(TAPEOUT)
      CALL         PRTOBLD(ALPHSTR,ALPHSTP,EXCPTNS,NEXCPTS,TOTALPH,
     % LTRSTRT,LTRSTOP,NUMSUB,RANGE,NSPACE,ENORM,DC,TAPEIN,RANDIN,LSPACE
     %,TAPEOUT,NEWFILE,ADD,CHANGE,DELETE)
      CALL PRTOTBL(LTRSTRT,LTRSTOP,TOTPRTO,.T.,.F.,.T.,.F.,NSPACE,
     %ENORM,DC,TAPEOUT,.F.,0)
      STOP
      END
      SUBROUTINE  ENORMFL (FLTRLTR,NORMLTR,NSPACE,DC)
      REAL FLTRLTR(NSPACE),NORMLTR(NSPACE)
      INTEGER NSPACE,DC
      INTEGER START,NVECTOR     $  REAL SUMSQRS
      START=2
      IF (DC.EQ.1) START=1
      SUMSQRS=0.0
      DO 100 NVECTOR=START,NSPACE
100   SUMSQRS=SUMSQRS+FLTRLTR(NVECTOR)**2
      SUMSQRS=SUMSQRS**.5
      IF (SUMSQRS.EQ.0.0) SUMSQRS=1.0
      DO 110 NVECTOR=START,NSPACE
110   NORMLTR(NVECTOR)=FLTRLTR(NVECTOR)/SUMSQRS
      IF (DC.EQ.0) NORMLTR(1)=1
      RETURN
      END
      SUBROUTINE GETFLTR (ALPHNUM,LTRNUM,LSPACE,NSPACE,TAPENUM,RANDOM)
      INTEGER ALPHNUM,LTRNUM,LSPACE,NSPACE,TAPENUM
      LOGICAL RANDOM
      INTEGER RECFLTR
      REAL FLTRLTR(180),PROTO(180)
      COMMON/PROFLTR/FLTRLTR,PROTO,RECFLTR
      INTEGER RECRQD,RECSKIP,KEY
      DATA RECFLTR/1/
      NUMWRDS=LSPACE+2
      IF  (.NOT.RANDOM) GO TO 95
      KEY=(ALPHNUM-1)*26+LTRNUM
      CALL READMS(TAPENUM,FLTRLTR,NUMWRDS,KEY)
      GO TO 150
95    CONTINUE
      RECRQD=(ALPHNUM-1)*26+LTRNUM
      RECSKIP=IABS(RECRQD-RECFLTR)
      IF (RECRQD-RECFLTR) 100,140,120
100   DO 110 I=1,RECSKIP
110   BACKSPACE TAPENUM
      GO TO 140
120   DO 130 I=1,RECSKIP
130   READ (TAPENUM)
140   READ (TAPENUM) (FLTRLTR(I),I=1,NUMWRDS)
```

```
      RECFLTR=RECROD+1
  150 CALL REDUCE(FLTRLTR,LSPACE,FLTRLTR,NSPACE)
      RETURN
      END
      SUBROUTINE PREPFLR(FORTAPE)
      INTEGER FORTAPE
      INTEGER FINDEX(3901)
      CALL OPENMS(FORTAPE,FINDEX,3901,0)
      RETURN
      END
      SUBROUTINE PREPRTO(PROTAPE)
      INTEGER PROTAPE
      INTEGER PINDEX(288)
      CALL OPENMS(PROTAPE,PINDEX,288,0)
      RETURN
      END
      SUBROUTINE  PRTOBLD(ALPHSTR,ALPHSTP,EXCPTNS,NEXCPTS,TOTALPH,
     % LTRSTRT,LTRSTOP,NUMSUB,RANGE,NSPACE,ENORM,DC,TAPEIN,RANDIN,.SPACE
     %,TAPEOUT,NEWFILE,ADD,CHANGE,DELETE)
      INTEGER ALPHSTR,ALPHSTP,EXCPTNS(20),NEXCPTS ,TOTALPH,LTRSTRT,
     % LTRSTOP,NUMSUB,RANGE(2,10),NSPACE(10),TAPEIN,TAPEOUT,ENORM,DC
     %,LSPACE
      LOGICAL RANDIN,NEWFILE,ADD,CHANGE,DELETE
      INTEGER OLDLTR,LTRNUM,CNTALPH(27),PROTNUM,SUBLTRS,KEY,PSPACE,
     % NCOL,NROW,ALPHNUM,                  WIDTH,NWORD,NVECTOR,NEXISTS
      REAL PROTOS(169,10)
      INTEGER MTABLE(27,5),TABLE(15,10),ETABLE(260,14)
      COMMON/TABLES/ MTABLE,TABLE,ETABLE
      INTEGER RECFLTR
      REAL FLTRLTR(180),PROTO(180)
      COMMON/PROFLTR/FLTRLTR,PROTO,RECFLTR
      DATA MTABLE/135*0/
      DATA LTRINDX,OLDLTR,CNTALPH/27,27,27*0/
      OLDLTR=LTRSTRT
      IF (NEWFILE) GO TO 90
      CALL READMS(TAPEOUT,MTABLE,135,1)
      IF (ADD.OR.CHANGE) GO TO 90
      IF (.NOT.DELETE) GO TO 90
      DO 80 LTRNUM=LTRSTRT,LTRSTOP
      DO 80  NCOL=1,5
   80 MTABLE(LTRNUM,NCOL)=0
   90 DO 230 LTRNUM=LTRSTRT,LTRSTOP
      IF (CNTALPH(OLDLTR).GE.MTABLE(OLDLTR,3)) GO TO 105
C     IF THE OLDLTR EXISTS SAVE THE OLDLTR PROTOTYPE INFORMATION
      CALL WRITMS(TAPEOUT,TABLE,NUMWRDS,KEY)
      DO 100 PROTNUM=1,SUBLTRS
  100 IF (TABLE(1,PROTNUM).EQ.1) CALL WRITMS(TAPEOUT,PROTOS(1,PROTNUM)
     %,TABLE(8,PROTNUM),TABLE(2,PROTNUM))
  105 IF (CNTALPH(LTRNUM).EQ.0) GO TO 120
C     IF THE NEW LETTER ALREADY EXISTS BUT IS NOT COMPLETE READ IT
C     INTO CORE TO BE WORKED ON
      KEY=MTABLE(LTRNUM,2)
      SUBLTRS=MTABLE(LTRNUM,3)
      NUMWRDS=SUBLTRS*15
      CALL READMS(TAPEOUT,TABLE,NUMWRDS,KEY)
      DO 115 PROTNUM=1,SUBLTRS
      IF (TABLE(1,PROTNUM).EQ.0) GO TO 110
      CALL READMS(TAPEOUT,PROTOS(1,PROTNUM),TABLE(8,PROTNUM),
     % TABLE(2,PROTNUM))
      GO TO 115
  110 DO 112 NROW=1,PSPACE
  112 PROTOS(NROW,PROTNUM)=0
  115 CONTINUE
      GO TO 160
  120 CONTINUE
```

```
      MTABLE(LTRNUM,1)=1
      MTABLE(LTRNUM,2)=KEY=LTRNUM+1
      MTABLE(LTRNUM,3)=SUBLTRS=NUMSUB
      NUMWRDS=SUBLTRS*15
      MTABLE(LTRNUM,5)=TOTALPH
      DO 150 NCOL=1,SUBLTRS
      LTRINDX=LTRINDX+1
      TABLE(1,NCOL)=0
      TABLE(2,NCOL)=LTRINDX
      TABLE(3,NCOL)=LTRNUM
      TABLE(4,NCOL)=NCOL
      TABLE(5,NCOL)=RANGE(1,NCOL)
      TABLE(6,NCOL)=RANGE(2,NCOL)
      TABLE(7,NCOL)=RANGE(2,NCOL)-RANGE(1,NCOL)+1
      TABLE(8,NCOL)=PSPACE=NSPACE(NCOL)
      TABLE(9,NCOL)=ENORM
      TABLE(10,NCOL)=DC
      DO 130 NROW=11,15
  130 TABLE(NROW,NCOL)=0
      DO 140 NROW=1,PSPACE
  140 PROTOS(NROW,NCOL)=0.0
  150 CONTINUE
  160 CONTINUE
      DO 200 ALPHNUM=ALPHSTR,ALPHSTP
      IF (NEXCPTS.EQ.0) GO TO 175
      DO 170 NROW =1,NEXCPTS
  170 IF (ALPHNUM.EQ.EXCPTNS(NROW )) GO TO 200
  175 CNTALPH(LTRNUM)=CNTALPH(LTRNUM)+1
       CALL GETFLTR(ALPHNUM,LTRNUM,LSPACE,LSPACE,TAPEIN,RANDIN)
      WIDTH=FLTRLTR(LSPACE+1)
      DO 190 PROTNUM=1,SUBLTRS
      IF ((WIDTH.LT.TABLE(5,PROTNUM)).OR.(WIDTH.GT.TABLE(6,PROTNUM)))
     % GO TO 190
      PSPACE=TABLE(8,PROTNUM)
      TABLE(12,PROTNUM)=TABLE(12,PROTNUM)+1
      MASKER=MASK(1)
      NWORD=60-MOD(ALPHNUM-1,60)
      MASKER=SHIFT(MASKER,NWORD)
      NWORD=(ALPHNUM-1)/60+13
      TABLE(NWORD,PROTNUM)=MASKER.OR.TABLE(NWORD,PROTNUM)
      CALL REDUCE(FLTRLTR,LSPACE,PROTO,PSPACE)
C#    XXXX=99.$ IF(ENORM.EQ.1) CALL ENORMFL(PROTO,PROTO,PSPACE,DC)
C#    XXXX=99$ CALL LEAR1(LTRNUM,PROTNUM,ENORM,DC,PROTO,PSPACE)
      DO 180 NVECTOR=1,PSPACE
  180 PROTOS(NVECTOR,PROTNUM)=PROTO(NVECTOR)+PROTOS(NVECTOR,PROTNUM)
      TABLE(1,PROTNUM)=1
  190 CONTINUE
  200 CONTINUE
      OLDLTR=LTRNUM
      IF(CNTALPH(LTRNUM).GE.TOTALPH) GO TO 215
      GO TO 230
  205 NEXISTS=0
      DO 220 PROTNUM =1,SUBLTRS
      IF (TABLE(1,PROTNUM).EQ.0) GO TO 220
      NEXISTS=NEXISTS+1
      PSPACE=TABLE(8,PROTNUM)
      DO 210 NVECTOR=1,PSPACE
  210 PROTOS(NVECTOR,PROTNUM)=PROTOS(NVECTOR,PROTNUM)/TABLE(12,PROTNUM)
      XXXX=99.$ IF(ENORM.EQ.1) CALL ENORMFL(PROTOS(1,PROTNUM),
     %  PROTOS(1,PROTNUM),PSPACE,DC)  $XXXX=99.
      TABLE(11,PROTNUM)=TABLE(12,PROTNUM)
C#    XXXX=99$ CALL LEAR2(PROTOS(1,PROTNUM),PROTNUM,PSPACE,DC,ENORM)
      CALL WRITMS(TAPEOUT,PROTOS(1,PROTNUM),PSPACE,TABLE(2,PROTNUM))
  220 CONTINUE
      MTABLE(LTRNUM,4)=NEXISTS
      CALL WRITMS(TAPEOUT,TABLE,NUMWRDS,KEY)
```

```
      CALL WRITMS(TAPEOUT,MTABLE,135,1)
C?    XXXX=99$CALL PRTOSTX(LTRSTRT,LTRSTOP,LSPACE,TAPEIN,RANDIN,.F.,
C?   1  .F.,.F.,NSPACE,ENORM,DC,TAPEOUT,.F.,.F.,.F.,1,1)
      RETURN
      END
      SUBROUTINE  PRTOTBL(LTRSTRT,LTRSTOP,TOTPRTO,PRINTIT,SUMMARY,
     %SKIPAGE,
     %OVERIDE,OSPACE,OENORM,ODC,TAPENUM,NEWTAPE,TAPETMP)
      INTEGER LTRSTRT,LTRSTOP,TOTPRTO,TAPENUM,TAPETMP,OSPACE(10)
     %,OENORM,ODC
      LOGICAL PRINTIT,SKIPAGE,OVERIDE,NEWTAPE,SUMMARY
      INTEGER LETTERS(26),MASKER,LTRNUM,KEY,SUBLTRS,PROTNUM,
     % NSPACE,PSPACE,NCOL,NROW,NALPHS,NTEMP,ALPHNUM,ALPHSTR,ALPHSTP,
     % ALPH(150),ENORM,DC,NINDEX(300)
      INTEGER MTABLE(27,5),TABLE(15,10),ETABLE(260,14)
      COMMON/TABLES/ MTABLE,TABLE,ETABLE
      INTEGER RECFLTR
      REAL FLTRLTR(180),PROTO(180)
      COMMON/PROFLTR/FLTRLTR,PROTO,RECFLTR
1     FORMAT(1H0)
  5   FORMAT (1H1)
10    FORMAT(1H ,41X,54("*"),/,1H ,41X,"CHARACTERISTICS OF PROTOTYPE FIL
     %E "I2", LETTERS "I2" TO "I2,/,1H ,41X,54("*"))
11    FORMAT(1H0,42("*")/1H ,"LTR SUBLTR RANGE SPACE ENORM DC TERM #ALPH
     % "/1H ,42("*"))
12    FORMAT (1H ,"CLASS #"I2", LETTER "A1"  *** DOES NOT EXIST***")
  15  FORMAT (=("*"))
  20  FORMAT (1H ,"CLASS #"I2", LETTER "A1)
25    FORMAT(1H0,"TOTAL # OF ALPHABETS="I3
     %       ".  # OF (POSSIBLE) SUBCLASSES="I2,/" RANGE OF(POSSIBLE) "
     %"SUBCLASSES IN ASCENDING ORDER IS:"I2,"-"I2,9(",",I2,"-"I2))
30    FORMAT(1H ,"# OF (ACTUAL) PROTOTYPES ="I2". CHARACTERISTICS FOR "
     %"THESE PROTOTYPES ARE SHOWN BELOW."/ )
35    FORMAT(1H ,"LTR SUBLTR RANGE SPACE ENORM DC TERM #ALPH  ALPHABETS
     %IN THE PROTOTYPE")
40    FORMAT(1H ,1X,A1,3X,I2,3X,I2,"-",I2,3X,I3,3X,A3,4X,A3,4X,I3,3X,
     % 22(I3,",")/,6(1H ,44X,22(I3,",")/))
41    FORMAT(1H ,1X,A1,3X,I2,3X,I2,"-",I2,3X,I3,3X,A3,4X,A3,4X,I3,3X)
45    FORMAT(1H ,42("-"))
50    FORMAT(1H ,1X,A1,3X,I2,3X,I2,"-",I2,3X,I3,3X,A3,4X,A3,4X,I3,3X,
     % "THIS SUBLTR DOES NOT EXIST, NO ALPHABETS IN THE RANGE")
      DATA LETTERS/"A","B","C","D","E","F","G","H","I","J","K","L","M",
     %"N","O","P","Q","R","S","T","U","V","W","X","Y","Z"/
      TOTPRTO=0
      IF (OVERIDE.AND.NEWTAPE) CALL OPENMS(TAPETMP,NINDEX,288,0)
      CALL READMS(TAPENUM,MTABLE,135,1)
      MASKER=MASK(1)
      IF (SKIPAGE) WRITE 5
      IF (PRINTIT.AND.SUMMARY) PRINTIT=.F.
      IF (PRINTIT.OR.SUMMARY) WRITE 10,(TAPENUM,LTRSTRT,LTRSTOP)
      IF (SUMMARY) WRITE 11
      DO 160 LTRNUM=LTRSTRT,LTRSTOP
      IF (.NOT.PRINTIT) GO TO 85
      WRITE 1
      WRITE 15,(20)
      IF (MTABLE(LTRNUM,1).EQ.1) GO TO 80
      WRITE 12,(LTRNUM,LETTERS(LTRNUM))
      WRITE 15,(20)
      GO TO 150
80    WRITE 20,(LTRNUM,LETTERS(LTRNUM))
      WRITE 15,(20)
85    IF (MTABLE(LTRNUM,1).EQ.0) GO TO 160
      KEY=MTABLE(LTRNUM,2)
      SUBLTRS=MTABLE(LTRNUM,3)
```

```
      SUBLTRS=MTABLE(LTRNUM,3)
      NUMWRDS=SUBLTRS*15
      CALL READMS(TAPENUM,TABLE,NUMWRDS,KEY)
      IF (.NOT.PRINTIT) GO TO 90
      WRITE 25,(MTABLE(LTRNUM,5),
     %          SUBLTRS,(TABLE( 5,NCOL),TABLE(6,NCOL),NCOL=1,SUBLTRS))
      WRITE 30, (MTABLE(LTRNUM,4))
      WRITE 15,(71)
      WRITE 35
      WRITE 15,(71)
   90 CONTINUE
      DO 150 PROTNUM=1,SUBLTRS
       IF (TABLE(1,PROTNUM).EQ.0) GO TO 142
      TOTPRTO=TOTPRTO +1
      IF (.NOT. OVERIDE) GO TO 100
      PSPACE=TABLE(8,PROTNUM)
      TABLE(8,PROTNUM)=NSPACE=OSPACE(PROTNUM)
      TABLE(9,PROTNUM)=OENORM
      TABLE(10,PROTNUM)=ODC
      IF (.NOT.NEWTAPE) GO TO 100
      KEY=TABLE(2,PROTNUM)
      CALL READMS(TAPENUM,PROTO,PSPACE,KEY)
      CALL REDUCE (PROTO,PSPACE,PROTO,NSPACE)
      IF (OENORM.EQ.1) CALL ENORMFL(PROTO,PROTO,NSPACE,ODC)
      CALL WRITMS(TAPETMP,PROTO,NSPACE,KEY)
  100 CONTINUE
      DO 110 NCOL=2,15
  110 ETABLE(TOTPRTO ,NCOL-1)=TABLE(NCOL,PROTNUM)
      IF (SUMMARY) GO TO 142
      IF (.NOT.PRINTIT) GO TO 150
      NALPHS=0
      DO 140 NWRDS=1,3
      NTEMP=TABLE(12+NWRDS,PROTNUM)
      ALPHSTR= (NWRDS-1)*60+1
      ALPHSTP=ALPHSTR+59
      IF (ALPHSTP.GT.150) ALPHSTP=150
      DO 130 ALPHNUM= ALPHSTR,ALPHSTP
      IF (NTEMP.AND.MASKER)  125,127
  125 NALPHS=NALPHS+1
      ALPH(NALPHS)=ALPHNUM
  127 NTEMP=SHIFT(NTEMP,1)
  130 CONTINUE
  140 CONTINUE
  142 NCOL=TABLE(12,PROTNUM)
      ENORM=DC="OUT"
      IF(TABLE(9,PROTNUM).EQ.1) ENORM="IN "
      IF(TABLE(10,PROTNUM).EQ.1) DC="IN "
      IF (TABLE(1,PROTNUM).EQ.0) 145,147
  145 WRITE 50,(LETTERS(LTRNUM),PROTNUM,TABLE(5,PROTNUM),TABLE(6,PROTNUM
     %),TABLE(8,PROTNUM),ENORM,DC,NCOL)
      IF (SUMMARY) 150,149
  147 IF (.NOT.SUMMARY) GO TO 148
      WRITE 41,(LETTERS(LTRNUM),PROTNUM,TABLE(5,PROTNUM),TABLE(6,PROTNUM
     %),TABLE(8,PROTNUM),ENORM,DC,NCOL)
      GO TO 150
  148 WRITE 40,(LETTERS(LTRNUM),PROTNUM,TABLE(5,PROTNUM),TABLE(6,PROTNUM
     %),TABLE(8,PROTNUM),ENORM,DC,NCOL   ,(ALPH(NROW),NROW=1,NALPHS))
  149 WRITE 45
  150 CONTINUE
  160 CONTINUE
      RETURN
      END
      SUBROUTINE REDUCE (INFLTR,INSPACE,OUTFLTR,OTSPACE)
      INTEGER INSPACE,OTSPACE
      REAL            INFLTR(INSPACE),OUTFLTR(OTSPACE)
      INTEGER INHRMNC,OUTHRMC,SKIP,DOWN,OUTVCTR,INVCTR,START,STOP
```

```
      DO 85 NVECTOR=1,OTSPACE
85    OUTFLTR(NVECTOR)=INFLTR(NVECTOR)
      GO TO 125
90    CONTINUE
      INHRMNC= INSPACE**.5   $  OUTHRMC=OTSPACE **.5
      SKIP=(INHRMNC-OUTHRMC)*2
      DOWN=OUTHRMC/2+1
      OUTFLTR(1)=INFLTR(1)
      IF (OTSPACE.EQ.1) RETURN
      DO 100 OUTVCTR=2,OUTHRMC
100   OUTFLTR(OUTVCTR)=INFLTR(OUTVCTR)
      INVCTR=START=OUTHRMC+1
      DO 120 NROW=2,DOWN
      INVCTR=INVCTR+SKIP
      STOP=START+OUTHRMC*2-1
      DO 110 OUTVCTR=START,STOP
      OUTFLTR(OUTVCTR)=INFLTR(INVCTR)
      INVCTR=INVCTR+1
110   CONTINUE
      START=STOP+1
120   CONTINUE
125   CONTINUE
      DO 130 I=1,4
130   OUTFLTR(OTSPACE+I)=INFLTR(INSPACE+I)
      RETURN
      END
      SUBROUTINE PRTOSTX(LTRSTRT,LTRSTOP,LSPACE,LTRTAPE,RANDOM,PRINTIT,
     % SKIPAGE,OVERIDE,OSPACE,OENORM,ODC,PROTAPE,COROLTN,HSTOGRM,BARS,
     % NDEV,NDIVS,WRATIO,HRATIO)
      INTEGER LTRSTRT, LTRSTOP,OSPACE(10),PROTAPE,LSPACE,NDEV,NDIVS,
     %LTRTAPE,OENORM,ODC
      REAL WRATIO,HRATIO
      LOGICAL PRINTIT,SKIPAGE,OVERIDE,COROLTN,HSTOGRM,BARS,RANDOM
      INTEGER MTABLE(27,5),TABLE(15,10),ETABLE(260,14)
      INTEGER RECFLTR
      REAL FLTRLTR(180),PROTO(180)
      COMMON/PROFLTR/FLTRLTR,PROTO,RECFLTR
      COMMON/TABLES/ MTABLE,TABLE,ETABLE
      INTEGER ALPHSTR,ALPHSTP,NWRDS,PROTNUM,NTEMP,TOTPRTO,ALPHNUM
     %,PENORM,PDC,TOTALPH,PSPACE,SUBLTR
      MASKER=MASK(1)
      CALL  PRTOTBL(LTRSTRT,LTRSTOP,TOTPRTO,PRINTIT,.F.,SKIPAGE,OVERIDE,
     %OSPACE,OENORM,ODC,PROTAPE,.F.,0)
      DO 120  PROTNUM=1,TOTPRTO
      LTRNUM =ETABLE(PROTNUM, 2)
      IF ((LTRNUM.LT.LTRSTRT ).OR.(LTRNUM.GT.LTRSTOP)) GO TO 120
      SUBLTR= ETABLE(PROTNUM,3)
      TOTALPH=ETABLE(PROTNUM,11)
      PSPACE= ETABLE (PROTNUM,7)
      PENORM=ETABLE(PROTNUM,8)
      PDC=ETABLE(PROTNUM,9)
      CALL READMS(PROTAPE,PROTO,PSPACE,ETABLE(PROTNUM,1))
      IF(IQ.NE.23)PRINT*," PROT ",PROTO $ IQ=23
      DO 110 NWRDS=1,3
      ALPHSTR= (NWRDS-1) *60+1
      ALPHSTP = ALPHSTR+59
      NTEMP=ETABLE(PROTNUM,11+NWRDS)
      DO 100 ALPHNUM= ALPHSTR,ALPHSTP
      IF(NTEMP.AND.MASKER) 95,97
C5    CALL STATS(ALPHNUM,ALPHNUM,TOTALPH,LTRNUM,SUBLTR,LSPACE,PSPACE,
CCCCC%CLTRTAPE,RANDOM,PENORM,PDC,COROLTN,HSTOGRM,BARS,NDEV,NDIVS,CCCCCCC
CCCCC%CPROTAPE,WRATIO,HRATIO)CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 95   CALL GETFLTR(ALPHNUM,LTRNUM,LSPACE,PSPACE,LTRTAPE,RANDOM)
      CALL STATX(FLTRLTR,PSPACE,TOTALPH)
97    NTEMP=SHIFT(NTEMP,1)
100   CONTINUE
```

```
      CONTINUE
  110 CONTINUE
      CALL MAHPAK(PROTO,PSPACE)
      CALL WRITMS(PROTAPE,PROTO,PSPACE,ETABLE(PROTNUM,1))
  120 CONTINUE
      RETURN
      END
      SUBROUTINE STATX(FVEC,NENT,NTOT)
C**********          INSERT TO PROTO BUILD  2/8/78     JRL
      COMMON/STX/ STNDDEV(81,3),FMEAN(81,3)
      DIMENSION FVEC(NENT),FT(2,81)
      IF (INUM.EQ.0) GOTO 60
      DO 50 J=1,169
   50 FT(1,J)=FT(2,J) = 0.0
   60 CONTINUE
      INUM=INUM+1
      DO  100 J=1,NENT
      CALL DOIT(FVEC(J),STNDDEV(J),FMEAN(J),FT(1,J),INUM)
  100 CONTINUE
      IF(INUM.EQ.NTOT) INUM = 0
      RETURN
      END
      SUBROUTINE DOIT(FIN,FSD,FM,FT,INUM)
C**********          INSERT TO PROTO BUILD  2/8/78     JRL
      DIMENSION FT(1)
      FT(1)=FT(1) + FIN**2
      FT(2)= FT(2) + FIN
      FM=FT(2)/INUM
      FSD = (FT(1)/INUM) - ((FT(2)/INUM)**2)
      FSD = SQRT(ABS(FSD))
      RETURN
      END
      SUBROUTINE MAHPAK(PROTO,PSPACE)
C**********          INSERT TO PROTO BUILD  2/8/78     JRL
      COMMON/STX/ STNDDEV(81,3),FMEAN(81,3)
      INTEGER PSPACE
      DIMENSION PROTO(PSPACE)
      IF(IQ.NE.23) PRINT*," FM ",FMEAN
      IF(IQ.NE.23) PRINT*," SDEV ",STNDDEV  $ IQ=23
      DO 20 I=1,PSPACE
   20 PROTO(I)=PACK(FMEAN(I),STNDDEV(I))
      RETURN
      END
      REAL FUNCTION PACK(A,B)
C**********          INSERT TO PROTO BUILD  2/8/78     JRL
      IA=A*1E5     $     IB = B*1E5
      PACK=(SHIFT(IA,30).AND.MASK(30)).OR..NOT.MASK(30).AND.IB
      IF(IA.LT.0) PACK=PACK.OR.MASK(1)
      IF(IB.LT.0) PACK = PACK.OR.SHIFT(MASK(1),30)
      RETURN
      END
      SUBROUTINE LEAR1(NEXT,ISET,NORM,IDC,FVEC,NDIM)
C**********  INSERT TO PRTOBLD 2/17/78  JRL
      COMMON/STX1/FT(2,81,3),INUM(3)
      DIMENSION FVEC(NDIM)
      COMMON/STX/SDEV(81,3),FM(81,3)
      DATA IGO/0/
      IF(NEXT.NE.IGO)GOTO 20
   10 CONTINUE
C**********  UPDATE ONE OF 3SPECIFIED MEANAND SDEV VECTORS
      INUM(ISET)=INUM(ISET)+1  $ IX=INUM(ISET)
      IF(NORM.EQ.1) CALL ENORMFL(FVEC,FVEC,NDIM,IDC)
      DO 15 J= 1,NDIM
      CALL DOIT(FVEC(J),SDEV(J,ISET),FM(J,ISET),FT(1,J,ISET),IX)
   15 CONTINUE
      RETURN
   20 CONTINUE
```

```
C**********  REINITIALIZE FOR EACH NEW SETOF 3 VECTOR STREAMS
      IGO=NEXT
      DO 30 J=1,81
      DO 30 K=1,3
      INUM(K)=0
 30   FT(1,J,K)=FT(2,J,K)=0.0
      GOTO 10
      END
      SUBROUTINE LEAR2(PROTO,ISET,PSPACE,DC,NORM)
C**********  INSERT TO PRTOBLD 2/17/78  JRL
      COMMON/STX1/FT(2,81,3),INUM(3)
      COMMON/STX/SD(81,3),FM(81,3)
      INTEGER PSPACE,START
      DIMENSION PROTO(1)
      START=2$ IF(DC.EQ.1) START=1
      IF(NORM.EQ.1)CALL ENORMFL(FM(1,ISET),FM(1,ISET),PSPACE,DC)
      DO 10 I=START,PSPACE
      SD(I,ISET)=FT(1,I,ISET)/INUM(ISET)-FM(I,ISET)**2
 10   SD(I,ISET)=SQRT(ABS(SD(I,ISET)))
      IF(IQ.NE.23)PRINT*," PR ",(PROTO(K),K=1,PSPACE)
      IF(IQ.NE.23)PRINT*," FM "  , (FM(K,ISET),K=1,PSPACE)
      IF(IQ.NE.23) PRINT*," SD ",(SD(K,ISET),K=1,PSPACE)
      II=II+1 $ IF(II.GT.1) IQ=23
      DO 20 I=1,PSPACE
 20   PROTO(I)=PACK(FM(I,ISET),SD(I,ISET))
      RETURN
      END
```

Appendix E

CMPRSEN

```
      PROGRAM MAIN(INPUT,OUTPUT,TAPE10,TAPE50,TAPE31,TAPE55)
      INTEGER STRTCOL,STOPCOL,NSKIPS,SENTNUM,SENTAPE,TAPETMP,
     % OSPACE(10),OENORM,ODC,LTRSTRT,LTRSTOP,PROTAPE,TOPRTOS,SENTEMP
      LOGICAL OVERIDE,BETWEEN,WINDOWS,PRNTALL ,PRNTSUM,ADJUST
      REAL WRATIO,HRATIO
      INTEGER LWIDTH,HWIDTH,SKWIDTH
      LWIDTH=2   $  HWIDTH=32  $  SKWIDTH=2
      STRTCOL=1  $  STOPCOL=133 $NSKIPS=1  $  SENTNUM=4  $  SENTAPE=50
CDATA     NEXT LINE VARIES WHICH SENTENCE USED FOR DATA
      SENTNUM=1
      OVERIDE=.T.  $  OENORM=1  $  ODC=0  $  LTRSTRT=1  $  LTRSTOP=26
      PROTAPE=10  $  PRNTALL=.T.  $  BETWEEN=.T.  $  WINDOWS=.T.
      NBTWTOP=5
       TOPRTOS=5   $  TAPETMP=31  $  ADJUST=.T.  $  WRATIO=1.5
      HRATIO=1.5$ PRNTSUM=.F. $  SENTEMP=55
      DATA OSPACE/10*81/
      CALL PREPSEN(SENTAPE)
      CALL PREPRTO(PROTAPE)
      CALL       CMPRSEN(STRTCOL,STOPCOL,NSKIPS,SENTNUM,SENTAPE,OVERIDE,
     %OSPACE,OENORM,ODC,LTRSTRT,LTRSTOP,PROTAPE,PRNTALL,PRNTSUM,BETWEEN,
     %WINDOWS,TOPRTOS,TAPETMP,ADJUST,WRATIO,HRATIO,SENTEMP,LWIDTH
     % ,HWIDTH,SKWIDTH,NBTWTOP)
      STOP
      END
      SUBROUTINE CMPRSEN(STRTCOL,STOPCOL,NSKIPS,SENTNUM,SENTAPE,OVERIDE,
     %OSPACE,OENORM,ODC,LTRSTRT,LTRSTOP,PROTAPE,PRNTALL,PRNTSUM,BETWEEN,
     %WINDOWS,TOPRTOS,TAPETMP,ADJUST,WRATIO,HRATIO,SENTEMP,LWIDTH
     % ,HWIDTH,SKWIDTH,NBTWTOP)
      INTEGER STRTCOL,STOPCOL,NSKIPS,SENTNUM,SENTAPE,TAPETMP,
     % OSPACE(10),OENORM,ODC,LTRSTRT,LTRSTOP,PROTAPE,TOPRTOS,SENTEMP
      LOGICAL OVERIDE,BETWEEN,WINDOWS,PRNTALL ,PRNTSUM,ADJUST
      REAL WRATIO,HRATIO
      INTEGER PRNTCOL,LTRMASK,SUBMASK,DXMASK,NSPACE,ENORM,DC,
     % LSTPRTO,TOTPRTO,PROTNUM,NCOL,FLAG,TAPENUM,KEY,PSPACE,
     % PENORM,PDC,START,STOP,PRTOCNT,LASTROW,I,J,XFRMCOL,NSPLITS,NPAGE,
     % STRTROW,STOPROW,SUBLTR(52),LTR(52),DX(52),TEMP,LOW,HIGH,SUBCNT
     %,TOPNUM,LETTERS(26),ENDCOL,SINDEX(134)
     %,WMASK   ,WIDTH,LWIDTH,HWIDTH,SKWIDTH,LRANGE,HRANGE
      COMMON/LETTERS/LETTER,NUMLTRS,RECNUM
      INTEGER LETTER(32,32),NUMLTRS,RECNUM
      COMMON/RESULTS/RESULTS
      INTEGER RESULTS(52,133)
      COMMON/PROFLTR/FLTRLTR,PROTO,RECFLTR
      INTEGER RECFLTR
      REAL FLTRLTR(180),PROTO(180)
      COMMON/TABLES/ MTABLE,TABLE,ETABLE
      INTEGER MTABLE(27,5),TABLE(10,15),ETABLE(260,14)
      REAL WINDOW(180)
      EQUIVALENCE(WINDOW,FLTRLTR)
5     FORMAT(1HT)
10    FORMAT(1H1)
15    FORMAT(1H ,136("*"),/1H ,24X,"RESULTS OF ALGORITHM TO SOLVE THE SE
     %GMENTATION PROBLEM FOR NON-ISOLATED TEXT CHARACTERS"/
     %1H ,136("*"),///)
16    FORMAT(1H ,"SENTENCE "I3" COLUMN "I3" TO "I3" EVALUATED AGAINST"/
     %" PROTOTYPE SET "I2" LETTERS "I2" TO "I2" WITH A LETTER-TO-WINDOW
     %WIDTH RATIO OF "F4.2, /" AND A LETTER-TO-WINDOW HEIGHT RATIO OF "
     %F4.2,/," SENTENCE WINDOWS RANGE FROM "I2" TO "I2" COLUMNS WIDE"
     %" WITH SKIPS IN WINDOW SIZE OF "I2" COLUMNS",/," AND SKIPS BETWEEN
     % WINDOWS WITHIN THE SENTENCE OF "I2" SPACES"///)
17    FORMAT(1H ,"FINAL RESULTS OF")
18    FORMAT(1HO,"USER REQUESTED PRINTOUT OF THE CLOSEST "I3" PROTOTYPES
     % FOR THE FINAL DECISION MATRIX."/" HOWEVER, FOR THE PROTOTYPE FILE
```

```
      % FOR THE FINAL DECISION MATRIX.  HOWEVER, FOR THE PROTOTYPE FILE
      % AND RANGE OF LETTERS SPECIFIED, ONLY "I3" PROTOTYPES EXIST."/
      %" USER REQUEST FOR THE FINAL DECISION MATRIX REDUCED TO THE CLOSES
      %T "I3" PROTOTYPES"//)
   20  FORMAT(1H0,"PROTOTYPES IN THE MATRIX ARE: "26(A1,I1,1X),/,3
      %  (1H ,30X,26(A1,I1,1X)/))
   25  FORMAT(1H ,"RESULTS MATRIX SHOWING THE PROTOTYPE-TO-WINDOW "
      %/" DISTANCES FOR PROTOTYPES WHOSE WIDTH VARIATION ENABLES"
      %/" THE PROTOTYPE TO BE COMPARED TO WINDOWS OF WIDTH "I2" COLUMNS")
   26  FORMAT(1H ,"MATRIX IS SORTED BY PROTOTYPE.  PROTOTYPE-TO-WINDOW"
      % /" DISTANCE IS SHOWN AS IT VARIES CONTINUOUSLY OVER THE SENTENCE"
      % /" COLUMNS USING WINDOWS OF WIDTH "I2" BEGINNING AND ENDING AT"
      %/" THE COLUMNS SHOWN BY THE MATRIX LEFTMOST COLUMN")
   27  FORMAT(1H0,60X,"MATRIX "I2" OF "I2)
   30  FORMAT(        1H ,10("*"),=(9("*")),/" COLUMNS * ",
      % =(3X,A1,I1,4X),/,1H ,10("*"),=(9("*")))
   32  FORMAT(1H ,I3"-"I3," * "=(F7.3,2X))
   35  FORMAT(1H ,I3"-"I3"  * "=(A1,I1,"-",F5.3,1X))
   40  FORMAT(1H ,"MATRIX IS SORTED BY DISTANCE WITHIN THE SENTENCE"
      %/" COLUMNS.  THE "I2" TO "I2" CLOSEST IDENTIFYING PROTOTYPES ARE"
      %/" SHOWN FOR EACH WINDOW OF WIDTH "I2" BEGINNING AND ENDING AT"
      %/" THE SENTENCE COLUMN INDICATED BY THE LEFTMOST MATRIX COLUMN.")
   41  FORMAT(1H ,"MERGED RESULTS MATRIX SHOWING THE PROTOTYPE-TO-WINDOW"
      %/" DISTANCES FOR ALL PROTOTYEPS WHOSE RANGE VARIATION ENABLES THE"
      %/" PROTOTYPE TO BE COMPARED WITH WINDOWS OF WIDTH "I2" TO "I2
      %/ " COLUMNS.  THE MATRIX IS SORTED BY DISTANCE WITHIN THE SENTENCE
      %"/" COLUMNS TO SHOW THE TOP "I2" PROTOTYPE CHOICES REGARDLESS OF"
      %/" WINDOW SIZE.  ALL WINDOWS BEGIN AT THE SAME SENTENCE COLUMNS"
      %/" WHICH ARE SHOWN IN THE LEFTMOST MATRIX COLUMN")
   42  FORMAT(1H ,"MATRIX OF THE "I2" TO " I2" CLOSEST IDENTIFYING "
      %"PROTOTYPES FOR WINDOWS OF WIDTH "I2" SORTED BY DISTANCE")
   43  FORMAT(    1H 11 ("*"),=(9 ("*")),/,1H ,"SENTENCE * ",
      %=("GUESS "I2,1X),/,1H ,"COLUMNS  * "=("PROTO/DX "),/,1H 11 (" "),
      % =(9 ("*")))
   45  FORMAT(1H 2X,I3,2X" * ",=(A1,I1,"-",F5.3,1X))
   46  FORMAT(    1H 11 ("*"),=(16 ("*"))/,1H ,"SENTENCE * ",
      % =("   CHOICE #"I2"   "),/,1H ,"COLUMNS     "=("PROTO-WIDTH-DX  "),
      % /,1H ,11 ("*"),=(16 ("*")))
   50  FORMAT(1H ,2X,I3,6X,=(A1,I1,2X,I2,2X,F5.3,3X))
       DATA LETTERS/"A","B","C","D","E","F","G","H","I","J","K","L","M",
      %"N","O","P","Q","R","S","T","U","V","W","X","Y","Z"/
       LTRSIZE=32  $  LENGTH=133
       PRNTCOL=13  $  NRNTCOL=7
       LTRMASK=MASK(8)
       SUBMASK=SHIFT(MASK(8),52)
       WMASK=SHIFT(MASK(8),44)
       DXMASK=SHIFT(MASK(22),22)
       WRITE 10
       WRITE 15
       CALL OPENMS(SENTEMP,SINDEX,134,0)
       CALL GETSEN (32,SENTNUM,SENTAPE)
       CALL PRTOTBL(LTRSTRT,LTRSTOP,TOTPRTO,PRNTALL,PRNTSUM,.T.,OVERIDE,
      % OSPACE,OENORM,ODC,PROTAPE,OVERIDE,TAPETMP)
       TAPENUM=PROTAPE
       IF (OVERIDE) TAPENUM=TAPETMP
       TOPNUM=TOPRTOS
       IF (TOPNUM.LE.TOTPRTO) GO TO 80
       TOPNUM=TOTPRTO
   80  CONTINUE
C
C      INITIALIZE THE TOPNUM ROWS OF THE RESULTS ARRAY TO THE
C      LARGEST DISTANCE, LTR=Z AND SUBLTR=0
C
       I=26
       I=SHIFT(I,52)
       TEMP=2000000
       DO 70 NCOL=STRTCOL,STOPCOL,NSKIPS
```

```
      DO 70 NCOL=STRTCOL,STOPCOL,NSKIPS
      DO 70 NROW=1,TOPNUM
      RESULTS(NROW,NCOL)=I.OR.TEMP
 70   CONTINUE
C
C     SORT OF THE ETABLE BY WIDTH AND NSPACE
C
      IF (TOTPRTO.EQ.1) GO TO 123
      LSTPRTO=TOTPRTO-1
100   FLAG=0
      DO 120 PROTNUM=1,LSTPRTO
      IF ((ETABLE(PROTNUM,6).GT.ETABLE(PROTNUM+1,6)).OR.
     % ((ETABLE(PROTNUM,6).EQ.ETABLE(PROTNUM+1,6)).AND.
     %  (ETABLE(PROTNUM,7).GE.ETABLE(PROTNUM+1,7)))) GO TO 120
      DO 110 NCOL=1,14
      TEMP=ETABLE(PROTNUM,NCOL)
      ETABLE(PROTNUM,NCOL)=ETABLE(PROTNUM+1,NCOL)
      ETABLE(PROTNUM+1,NCOL)=TEMP
110   CONTINUE
      FLAG=1
120   CONTINUE
      IF (FLAG.EQ.1) GO TO 100
123   CONTINUE
C
C     BEGIN LOOPING ON THE PROTYPES CONTAINED IN THE EXIST TABLE
C     PRODUCED BY PRTOTBL
C
      WIDTH=0
      DC=0
      NSPACE=0
      ENORM=0
      PRTOCNT=0
      NLOOKS=1
      DO 380 LHWIDTH=1,NLOOKS
      NWIDTH=WIDTH-SKWIDTH*(LHWIDTH-1)
      DO 370 PROTNUM=1,TOTPRTO
      LRANGE=ETABLE(PROTNUM,4)
      HRANGE=ETABLE(PROTNUM,5)
      IF ((NWIDTH.LT.LRANGE).OR.(NWIDTH.GT.HRANGE)) GO TO 370
      KEY=ETABLE(PROTNUM,1)
      PSPACE=ETABLE(PROTNUM,7)
      PENORM=ETABLE(PROTNUM,8)
      PDC=ETABLE(PROTNUM,9)
      CALL READMS(TAPENUM,PROTO,PSPACE,KEY)
C
C     IF THE WIDTH OF THE PROTOTYPE CHANGES PRINTOUT THE INTERMEDIATE
C     RESULTS BEFORE CONTINUING WITH THE NEXT PROTOTYPE AND TRANSFORM TH
C     SENTENCE USING THE NEW WINDOW WIDTH
C     IF THE ENERGY NORMALIZATION OR THE DC TERM IN THE NORMALIZED
C     WINDOWS REQUIRES A CHANGE DUE TO A CHANGE IN THE PROTOTYPE
C     CHARACTERISTICS, THE ENTIRE SENTENCE MUST BE TRANSFORMED AGAIN
C
      IF ((WIDTH.NE.NWIDTH).OR.(PENORM.NE.ENORM).OR.(DC.NE.PDC))
     % CALL FORTSEN(LTRSIZE,NWIDTH,ADJUST,WRATIO,HRATIO,PSPACE,PENORM,
     %PDC,STRTCOL,STOPCOL,NSKIPS,SENTNUM,SENTEMP)
      IF ((WIDTH.NE.NWIDTH).OR.(PENORM.NE.ENORM).OR.(DC.NE.PDC))
     % NSPACE=PSPACE
C
C     IF THE SPACE OF THE PROTOTYPE CHANGES REDUCE THE SPACE OF THE
C     WINDOWS CONTAINED IN THE TEMPORARY SENTENCE FILE OF XFORMS
C
      IF (NSPACE.EQ.PSPACE) GO TO 350
      DO 340 XFRMCOL=STRTCOL,ENDCOL ,NSKIPS
      CALL READMS(SENTEMP,WINDOW,NSPACE,XFRMCOL)
      CALL REDUCE(WINDOW,NSPACE,WINDOW,PSPACE)
      CALL WRITMS(SENTEMP,WINDOW,PSPACE,XFRMCOL)
340   CONTINUE
```

```
340   CONTINUE
C
C     UPDATE ALL VARIABLE FOR THE CURRENT PROTOTYPE
C
      IF((PRTOCNT.NE.0).AND.(WIDTH.NE.NWIDTH )) GO TO 125
350   CONTINUE
      WIDTH=NWIDTH $ NSPACE=PSPACE $ ENORM=PENORM $ DC=PDC
      PRTOCNT=PRTOCNT+1
      NROW=PRTOCNT+TOPNUM
      ENDCOL=STOPCOL
      IF ((ENDCOL+WIDTH-1).GT.LENGTH) ENDCOL=LENGTH-WIDTH+1
C
C     STORE THE RESULTS OF THE PROTOTYPE AGAINST ALL THE SENTENCE
C     WINDOW TRANSFORMS
C
      DO 360 XFRMCOL=STRTCOL,ENDCOL,NSKIPS
      CALL READMS(SENTEMP,WINDOW,NSPACE,XFRMCOL)
      RESULTS(NROW,XFRMCOL)=EUCLID(PROTO,WINDOW,NSPACE,DC)*1000000
      I=SHIFT(ETABLE(PROTNUM,2),52)
      J=SHIFT(ETABLE(PROTNUM,3),44)
      I=I.OR.J
      J=SHIFT(WIDTH,36)
      I=I.OR.J
      RESULTS(NROW,XFRMCOL)=I.OR.RESULTS(NROW,XFRMCOL)
360   CONTINUE
C
C     IF THERE ARE BLANK COLUMNS IN THE RESULTS ARRAY, SET THE DISTANCE
C     TO 2.0 AND THE LETTER TO Z AND THE SUBLTR TO 0
C
      I=26
      I=SHIFT(I,52)
      TEMP=2000000
      DO 362 NCOL=XFRMCOL,STOPCOL,NSKIPS
      RESULTS(NROW,NCOL)=I.OR.TEMP
362   CONTINUE
365   IF (PROTNUM.EQ.TOTPRTO) GO TO 125
      GO TO 370
C
C     CONTINUE TO LOOP ON THE NEXT PROTOTYPE
C
125   STRTROW=TOPNUM+1
      STOPROW=STRTROW+PRTOCNT-1
      LASTROW=STOPROW-1
C
C     PRINTOUT OF THE PROTOTYPES OF WIDTH XX AND THE WINDOWS OF THE
C     SENTENCES THEY ARE BEING EVALUATED AGAINST
C
      IF (STRTROW.EQ.STOPROW) GO TO 295
C
C     SORT BY INCREASING LETTER PROTOTYPE FOR DISPLAY OF EACH LETTERS
C     PERFORMANCE BY SENTENCE COLUMN.  THIS IS THE FIRST OF THE OPTIONAL
C     PRINTOUTS FOR THE INTERMEDIATE RESULTS
C
155   FLAG=0
      DO 170 NROW=STRTROW,LASTROW
      IF ((LTRMASK.AND.RESULTS(NROW)).LE.(LTRMASK.AND.RESULTS(NROW+1)))
     % GO TO 170
      FLAG=1
      DO 160 XFRMCOL=STRTCOL,ENDCOL ,NSKIPS
      TEMP=RESULTS(NROW,XFRMCOL)
      RESULTS(NROW,XFRMCOL)=RESULTS(NROW+1,XFRMCOL)
      RESULTS(NROW+1,XFRMCOL)=TEMP
160   CONTINUE
170   CONTINUE
      IF (FLAG.EQ.1) GO TO 155
      DO 180 J= STRTROW,STOPROW
      LTR(J)=SHIFT(LTRMASK.AND.RESULTS(J,STRTCOL),8)
```

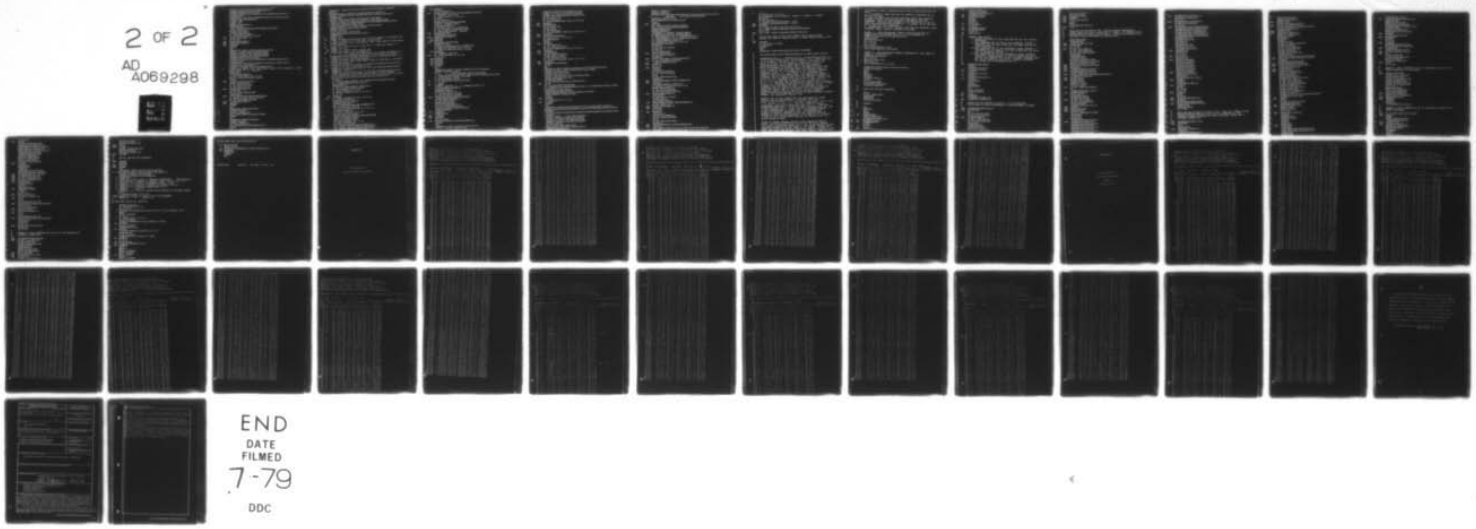
```
      LTR(J)=SHIFT(LTRMASK.AND.RESULTS(J,STRTCOL),8)
      SUBLTR(J)=SHIFT(SUBMASK.AND.RESULTS(J,STRTCOL),16)
180   CONTINUE
      WRITE 10
      WRITE 25,(WIDTH)
      WRITE 25,(WIDTH)
      NSPLITS=(PRTOCNT-1)/PRNTCOL+1
      START=STRTROW
      DO 210 NPAGE=1,NSPLITS
      STOP=START+PRNTCOL-1
      IF (NPAGE.EQ.NSPLITS) STOP=STOPROW
      SUBCNT=STOP-START+1
      IF (NPAGE.GT.1) WRITE 10
      WRITE 27,(NPAGE,NSPLITS)
      WRITE 20,(                LETTERS(LTR(J)),SUBLTR(J),J=START,STOP)
      WRITE 30,(SUBCNT,SUBCNT,(LETTERS(LTR    (J)),SUBLTR(J),
     % J=START,STOP),SUBCNT)
      DO 200 XFRMCOL=STRTCOL,ENDCOL ,NSKIPS
      DO 190 J=START,STOP
190   DX(J)=DXMASK.AND.RESULTS(J    ,XFRMCOL)
      WRITE 32,(XFRMCOL,XFRMCOL+WIDTH-1,
     % SUBCNT,(DX(J)/1000000.0,J=START,STOP))
200   CONTINUE
      START=STOP+1
210   CONTINUE
C
C     SORT BY DISTANCE WITHIN THE SENTENCE COLUMN FOR THE SECOND OUTPUT
C     OF THE INTERMEDIATE RESULTS
C
      CALL SORTIT(STRTCOL,ENDCOL,NSKIPS,STRTROW,STOPROW)
      START=STRTROW
      TEMP=NBTWTOP
      IF (NBTWTOP.GT.PRTOCNT) NBTWTOP=PRTOCNT
      NSPLITS=(NBTWTOP-1)/PRNTCOL+1
      LOW=1
      WRITE 10
      WRITE 25,(WIDTH)
      WRITE 40,(1,SUBCNT,WIDTH)
      WRITE 20,(LETTERS(LTR(J)),SUBLTR(J),J=STRTROW,STOPROW)
      DO 260 NPAGE=1,NSPLITS
      STOP=START+PRNTCOL-1
      IF (NPAGE.EQ.NSPLITS) STOP=STRTROW+NBTWTOP-1
      SUBCNT=STOP-START+1
      HIGH=LOW+SUBCNT-1
      IF (NPAGE.GT.1) WRITE 10
      WRITE 27,(NPAGE,NSPLITS)
      WRITE 42,(LOW,HIGH,WIDTH)
      WRITE 43,(SUBCNT,SUBCNT,(NROW,NROW=LOW,HIGH),SUBCNT,SUBCNT)
      DO 250 XFRMCOL=STRTCOL,ENDCOL ,NSKIPS
      DO 240 J=START,STOP
      LTR(J)=SHIFT(LTRMASK.AND.RESULTS(J    ,XFRMCOL),8)
      SUBLTR(J)=SHIFT(SUBMASK.AND.RESULTS(J    ,XFRMCOL),16)
      DX(J)=DXMASK.AND.RESULTS(J    ,XFRMCOL)
240   CONTINUE
      WRITE 35,(XFRMCOL,XFRMCOL+WIDTH-1,SUBCNT,(LETTERS(LTR(J)),
     % SUBLTR(J),DX(J)/1000000.0,J=START,STOP))
250   CONTINUE
      START=STOP+1
      LOW=HIGH+1
260   CONTINUE
      NBTWTOP=TEMP
295   CONTINUE
370   CONTINUE
380   CONTINUE
      RETURN
      END
      SUBROUTINE ARNGCEX(TEMP,NROWS,NCOLS,TAMP,NUMROWS,NUMCOLS)
```

```
      COMPLEX TEMP(NROWS,NCOLS),TAMP(NUMROWS,NUMCOLS)
      INTEGER NROWS,NCOLS,NUMROWS,NUMCOLS
      DO 100 NCOL=1,NUMCOLS
      DO 100 NROW=1,NUMROWS
      TAMP(NROW,NCOL)=TEMP(NROW,NCOL)
100   CONTINUE
      RETURN
      END
      SUBROUTINE  ENORMFL (FLTRLTR,NORMLTR,NSPACE,DC)
      REAL FLTRLTR(NSPACE),NORMLTR(NSPACE)
      INTEGER NSPACE,DC
      INTEGER START,NVECTOR     $  REAL SUMSQRS
      START=2
      IF (DC.EQ.1) START=1
      SUMSQRS=0.0
      DO 100 NVECTOR=START,NSPACE
100   SUMSQRS=SUMSQRS+FLTRLTR(NVECTOR)**2
      SUMSQRS=SUMSQRS**.5
      IF (SUMSQRS.EQ.0.0) SUMSQRS=1.0
      DO 110 NVECTOR=START,NSPACE
110   NORMLTR(NVECTOR)=FLTRLTR(NVECTOR)/SUMSQRS
      IF (DC.EQ.0) NORMLTR(1)=1
      RETURN
      END
      REAL FUNCTION EUCLID(FLTR1,FLTR2,NSPACE,DC)
      INTEGER NSPACE,DC
      REAL            FLTR1(NSPACE),FLTR2(NSPACE)
      REAL SUMSQRS  $  INTEGER START $ LOGICAL AMPSPEC
CDATA TO GET COMPLEX SPECTRUM -- AMPSPEC=FALSE
CDATA TO GET AMPLITUDE SPECTRUM -- AMPSPEC=TRUE
      AMPSPEC=.FALSE.
      SUMSQRS=0.0
      START=2
      IF (DC.EQ.1) START=1
      IF (AMPSPEC) GO TO 50
C THIS BRANCH FINDS EUCLEDIAN DISTANCE - COMPLEX SPECTRUM
      DO 25 NVECTOR=START,NSPACE
      SUMSQRS=SUMSQRS+(FLTR1(NVECTOR)-FLTR2(NVECTOR))**2
  25  CONTINUE
      GO TO 110
C THIS BRANCH FINDS EUCLEDIAN DISTANCE - AMPLITUDE SPECTRUM
  50  DO 100 NVECTOR=START,NSPACE,2
      AFLTR1=SQRT((FLTR1(NVECTOR)**2)+(FLTR1(NVECTOR+1)**2))
      AFLTR2=SQRT((FLTR2(NVECTOR)**2)+(FLTR2(NVECTOR+1)**2))
      SUMSQRS =SUMSQRS+(AFLTR1-AFLTR2)**2
100   CONTINUE
110   EUCLID=SUMSQRS**.5
      RETURN
      END
      SUBROUTINE FILTER(FORTLTR,NUMROWS,NUMCOLS,FLTRLTR,NSPACE)
      INTEGER NUMROWS,NUMCOLS,NSPACE
      COMPLEX FORTLTR(NUMROWS,NUMCOLS)
      REAL FLTRLTR(NSPACE)
      INTEGER NHRMNCS,NUMWRDS,RIGHT,LEFT,DOWN,NROW,NCOL
      NHRMNCS=NSPACE**.5
      RIGHT=NHRMNCS/2+1
      LEFT=NUMCOLS-RIGHT+2
      DOWN=RIGHT
      FLTRLTR(1)=REAL(FORTLTR(1,1))
      IF (NHRMNCS.EQ.1) RETURN
      NUMWRDS=0
      DO 100 NCOL=2,RIGHT
      NUMWRDS=NUMWRDS+2
      FLTRLTR(NUMWRDS)=REAL(FORTLTR(1,NCOL))
100   FLTRLTR(NUMWRDS +1 )=AIMAG(FORTLTR(1,NCOL))
      DO 220 NROW=2,DOWN
```

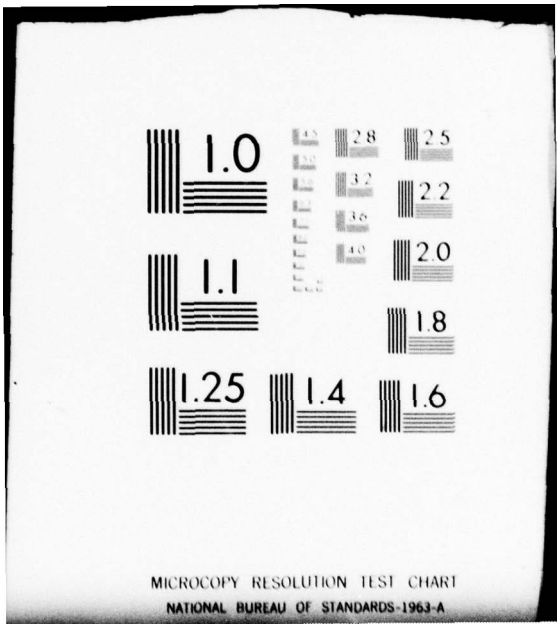
```
      DO 210 NCOL=LEFT,NUMCOLS
      NUMWRDS=NUMWRDS+2
      FLTRLTR(NUMWRDS  )=REAL(FORTLTR(NROW,NCOL))
      FLTRLTR(NUMWRDS+1)=AIMAG(FORTLTR(NROW,NCOL))
210   CONTINUE
      DO 200 NCOL=1,RIGHT
      NUMWRDS=NUMWRDS+2
      FLTRLTR(NUMWRDS  )=REAL(FORTLTR(NROW,NCOL))
      FLTRLTR(NUMWRDS+1)=AIMAG(FORTLTR(NROW,NCOL))
200   CONTINUE
220   CONTINUE
      RETURN
      END
      SUBROUTINE FORTSEN(LTRSIZE,WIDTH,ADJUST,WRATIO,HRATIO,NSPACE,
     % ENORM,DC,STRTCOL,STOPCOL,NSKIPS,SENTNUM,TAPENUM)
      INTEGER LTRSIZE,WIDTH,NSPACE,STRTCOL,STOPCOL,NSKIPS,SENTNUM,
     % TAPENUM,ENORM,DC
      LOGICAL ADJUST  $  REAL WRATIO,HRATIO
      INTEGER RECFLTR
      REAL FLTRLTR(180),PROTO(180)
      COMMON/PROFLTR/FLTRLTR,PROTO,RECFLTR
      COMMON/XFORMS/FORTLTR
      COMPLEX FORTLTR(64,64)
      INTEGER ROW,COL,NROW,NCOL,ENDCOL,XFRMCOL,LENGTH
      LENGTH=133
      ENDCOL=STOPCOL
      IF ((ENDCOL+WIDTH-1).GT.LENGTH) ENDCOL=LENGTH-WIDTH+1
      DO 130 XFRMCOL=STRTCOL,ENDCOL ,NSKIPS
      CALL MIDDLE(LTRSIZE,XFRMCOL,WIDTH)
      CALL XFORMIT(LTRSIZE,WIDTH,HEIGHT,ADJUST,WRATIO,HRATIO,NSPACE
     %,NUMROWS,NUMCOLS,0,0,.F.,.F.,.F.,.F.,SENTNUM)
      CALL FILTER(FORTLTR,NUMROWS,NUMCOLS,FLTRLTR,NSPACE)
      IF((ENORM.EQ.1).OR.(DC.EQ.1)) CALL ENORMFL(FLTRLTR,FLTRLTR,NSPACE,
     % DC)
      CALL WRITMS(TAPENUM  ,FLTRLTR,NSPACE,XFRMCOL)
130   CONTINUE
      RETURN
      END
      SUBROUTINE GETFLTR (ALPHNUM,LTRNUM,LSPACE,NSPACE,TAPENUM,RANDOM)
      INTEGER ALPHNUM,LTRNUM,LSPACE,NSPACE,TAPENUM
      LOGICAL RANDOM
      INTEGER RECFLTR
      REAL FLTRLTR(180),PROTO(180)
      COMMON/PROFLTR/FLTRLTR,PROTO,RECFLTR
      INTEGER RECRQD,RECSKIP,KEY
      DATA RECFLTR/1/
      NUMWRDS=LSPACE+4
      IF  (.NOT.RANDOM) GO TO 95
      KEY=(ALPHNUM-1)*26+LTRNUM
      CALL READMS(TAPENUM,FLTRLTR,NUMWRDS,KEY)
      GO TO 150
95    CONTINUE
      RECRQD=(ALPHNUM-1)*26+LTRNUM
      RECSKIP=IABS(RECRQD-RECFLTR)
      IF (RECRQD-RECFLTR) 100,140,120
100   DO 110 I=1,RECSKIP
110   BACKSPACE TAPENUM
      GO TO 140
120   DO 130 I=1,RECSKIP
130   READ (TAPENUM)
140   READ (TAPENUM) (FLTRLTR(I),I=1,NUMWRDS)
      RECFLTR=RECRQD+1
150   CALL REDUCE(FLTRLTR,LSPACE,FLTRLTR,NSPACE)
      RETURN
      END
      SUBROUTINE GETSEN(LTRSIZE,SENTNUM,TAPENUM)
```

AD-A069 298 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 9/2
SEGMENTATION OF TOUCHING ENGLISH LETTERS.(U)
MAR 79 R E BENTKOWSKI
UNCLASSIFIED AFIT/GE/EE/79-1 NL

2 OF 2
AD A069298



END
DATE FILMED
7-79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
      SUBROUTINE GETSEN(LTRSIZE,SENTNUM,TAPENUM)
      INTEGER LENGTH,LTRSIZE,SENTNUM,TAPENUM
      COMMON/SENTNCE/SAMPLE,HOLDIT
      INTEGER SAMPLE(32,133),HOLDIT(254),NUMCHAR,TITLE(30,4),
     % HOLDSEN(133)
      EQUIVALENCE (HOLDIT(1),NUMCHAR),(HOLDIT(2),TITLE(1,1)),
     % (HOLDIT(122),HOLDSEN(1))
      INTEGER TEMP
      LENGTH =133
      NUMWRDS=LENGTH+120+1
      MASKER=MASK(1)
       CALL READMS(TAPENUM,HOLDIT,NUMWRDS,SENTNUM)
      DO 130 NCOL=1,133
      TEMP=HOLDSEN(NCOL)
      DO 130 NROW=1,LTRSIZE
      IF (TEMP.AND.MASKER) 110,100
  100 SAMPLE(NROW,NCOL) =0
      GO TO 120
  110 SAMPLE(NROW,NCOL)=1
  120 TEMP=SHIFT(TEMP,1)
  130 CONTINUE
      RETURN
      END
      SUBROUTINE MIDDLE(LTRSIZE,XFRMCOL,WIDTH)
      INTEGER LENGTH,LTRSIZE,XFRMCOL,WIDTH
      COMMON/LETTERS/LETTER,NUMLTRS,RECNUM
      INTEGER LETTER(32,32),NUMLTRS,RECNUM
      COMMON/SENTNCE/SAMPLE,HOLDIT
      INTEGER SAMPLE(32,133),HOLDIT(254),NUMCHAR,TITLE(30,4),
     % HOLDSEN(133)
      EQUIVALENCE (HOLDIT(1),NUMCHAR),(HOLDIT(2),TITLE(1,1)),
     % (HOLDIT(122),HOLDSEN(1))
      INTEGER STOP,NROW,NCOL,LEDGE,REDGE,TEMP
    1 FORMAT(1H0,"WIDTH ("I3") PLUS START COLUMN ("I3") EXCEEDS THE SENT
     %ENCE LENGTH OF "I3" COLUMNS")
      LENGTH=133
      STOP=XFRMCOL+WIDTH-1
      IF (STOP.LE.LENGTH) GO TO 100
      WRITE 1,(WIDTH,XFRMCOL,LENGTH)
      RETURN
  100 LEDGE=(LTRSIZE-WIDTH)/2
      REDGE=LEDGE+WIDTH+1
      DO 140 NROW=1,LTRSIZE
      DO 110 NCOL=1,LEDGE
  110 LETTER(NROW,NCOL)=0
      TEMP=LEDGE
      DO 120 NCOL=XFRMCOL,STOP
      TEMP=TEMP+1
  120 LETTER(NROW,TEMP)=SAMPLE(NROW,NCOL)
      IF (REDGE.GT.LTRSIZE) GO TO 140
      DO 130 NCOL=REDGE,LTRSIZE
  130 LETTER(NROW,NCOL)=0
  140 CONTINUE
      RETURN
      END
      SUBROUTINE PREPRTO(PROTAPE)
      INTEGER PROTAPE
      INTEGER PINDEX(288)
      CALL OPENMS(PROTAPE,PINDEX,288,0)
      RETURN
      END
      SUBROUTINE PREPSEN(SENTAPE)
      INTEGER SENTAPE
      INTEGER SINDEX(244)
      CALL OPENMS(SENTAPE,SINDEX,244,0)
      RETURN
      END
```

```
      SUBROUTINE  PRTOTBL(LTRSTRT,LTRSTOP,TOTPRTO,PRINTIT,SUMMARY,
     %SKIPAGE,
     %OVERIDE,OSPACE,OENORM,OOC,TAPENUM,NEWTAPE,TAPETMP)
      INTEGER LTRSTRT,LTRSTOP,TOTPRTO,TAPENUM,TAPETMP,OSPACE(10)
     %,OENORM,OOC
      LOGICAL PRINTIT,SKIPAGE,OVERIDE,NEWTAPE,SUMMARY
      INTEGER LETTERS(26),MASKER,LTRNUM,KEY,SUBLTRS,PROTNUM,
     % NSPACE,PSPACE,NCOL,NROW,NALPHS,NTEMP,ALPHNUM,ALPHSTR,ALPHSTP,
     % ALPH(150),ENORM,OC,NINDEX(300)
      INTEGER MTABLE(27,5),TABLE(15,10),ETABLE(260,14)
      COMMON/TABLES/ MTABLE,TABLE,ETABLE
      INTEGER RECFLTR
      REAL FLTRLTR(180),PROTO(180)
      COMMON/PROFLTR/FLTRLTR,PROTO,RECFLTR
1     FORMAT(1H0)
5     FORMAT (1H1)
10    FORMAT(1H ,41X,54("*"),/,1H ,41X,"CHARACTERISTICS OF PROTOTYPE FIL
     %E "I2", LETTERS "I2" TO "I2,/,1H ,41X,54("*"))
11    FORMAT(1H0,42("*")/1H ,"LTR SUBLTR RANGE SPACE ENORM OC TERM #ALPH
     % "/1H ,42("*"))
12    FORMAT (1H ,"CLASS #"I2", LETTER "A1"  *** DOES NOT EXIST***")
15    FORMAT (=("*"))
20    FORMAT (1H ,"CLASS #"I2", LETTER "A1)
25    FORMAT(1H0,"TOTAL # OF ALPHABETS="I3
     %       ",  # OF (POSSIBLE) SUBCLASSES="I2,/" RANGE OF(POSSIBLE) "
     %"SUBCLASSES IN ASCENDING ORDER IS:"I2,"-"I2,9(",",I2,"-"I2))
30    FORMAT(1H ,"# OF (ACTUAL) PROTOTYPES ="I2". CHARACTERISTICS FOR "
     %"THESE PROTOTYPES ARE SHOWN BELOW."/ )
35    FORMAT(1H ,"LTR SUBLTR RANGE SPACE ENORM OC TERM #ALPH  ALPHABETS
     %IN THE PROTOTYPE")
40    FORMAT(1H ,1X,A1,3X,I2,3X,I2,"-",I2,3X,I3,3X,A3,4X,A3,4X,I3,3X,
     % 22(I3,",")/,6(1H ,44X,22(I3,",")/))
41    FORMAT(1H ,1X,A1,3X,I2,3X,I2,"-",I2,3X,I3,3X,A3,4X,A3,4X,I3,3X)
45    FORMAT(1H ,42("-"))
50    FORMAT(1H ,1X,A1,3X,I2,3X,I2,"-",I2,3X,I3,3X,A3,4X,A3,4X,I3,3X,
     % "THIS SUBLTR DOES NOT EXIST, NO ALPHABETS IN THE RANGE")
      DATA LETTERS/"A","B","C","D","E","F","G","H","I","J","K","L","M",
     %"N","O","P","Q","R","S","T","U","V","W","X","Y","Z"/
      TOTPRTO=0
      IF (OVERIDE.AND.NEWTAPE) CALL OPENMS(TAPETMP,NINDEX,288,0)
      CALL READMS(TAPENUM,MTABLE,135,1)
      MASKER=MASK(1)
      IF (PRINTIT.AND.SUMMARY) PRINTIT=.F.
      DO 160 LTRNUM=LTRSTRT,LTRSTOP
      IF (.NOT.PRINTIT) GO TO 85
      IF (MTABLE(LTRNUM,1).EQ.1) GO TO 80
      GO TO 150
80    CONTINUE
85    IF (MTABLE(LTRNUM,1).EQ.0) GO TO 160
      KEY=MTABLE(LTRNUM,2)
      SUBLTRS=MTABLE(LTRNUM,3)
      NUMWRDS=SUBLTRS*15
      CALL READMS(TAPENUM,TABLE,NUMWRDS,KEY)
      DO 150 PROTNUM=1,SUBLTRS
       IF (TABLE(1,PROTNUM).EQ.0) GO TO 150
      TOTPRTO=TOTPRTO +1
      IF (.NOT. OVERIDE) GO TO 100
      PSPACE=TABLE(8,PROTNUM)
      TABLE(8,PROTNUM)=NSPACE=OSPACE(PROTNUM)
      TABLE(9,PROTNUM)=OENORM
      TABLE(10,PROTNUM)=OOC
      IF (.NOT.NEWTAPE) GO TO 100
      KEY=TABLE(2,PROTNUM)
      CALL READMS(TAPENUM,PROTO,PSPACE,KEY)
      CALL REDUCE (PROTO,PSPACE,PROTO,NSPACE)
      CALL WRITMS(TAPETMP,PROTO,NSPACE,KEY)
```

```
      CALL WRITE(TTY,PROMPT,PROMPTL,NOECHO,ERR)
  100 CONTINUE
      DO 110 NCOL=2,15
  110 ETABLE(TOTPRTO ,NCOL-1)=TABLE(NCOL,PROTNUM)
      IF (SUMMARY) GO TO 142
      IF (.NOT.PRINTIT) GO TO 150
      NALPHS=0
      DO 140 NWRDS=1,3
      NTEMP=TABLE(12+NWRDS,PROTNUM)
      ALPHSTR= (NWRDS-1)*60+1
      ALPHSTP=ALPHSTR+59
      IF (ALPHSTP.GT.150) ALPHSTP=150
      DO 130 ALPHNUM= ALPHSTR,ALPHSTP
      IF (NTEMP.AND.MASKER)  125,127
  125 NALPHS=NALPHS+1
      ALPH(NALPHS)=ALPHNUM
  127 NTEMP=SHIFT(NTEMP,1)
  130 CONTINUE
  140 CONTINUE
  142 NCOL=TABLE(12,PROTNUM)
      ENORM="OUT"
      IF(TABLE(9,PROTNUM).EQ.1) ENORM="IN "
      IF(TABLE(10,PROTNUM).EQ.1) DC="IN "
      IF (TABLE(1,PROTNUM).EQ.0) 145,147
  145 CONTINUE
      IF (SUMMARY) 150,149
  147 IF (.NOT.SUMMARY) GO TO 148
      GO TO 150
  148 CONTINUE
  149 CONTINUE
  150 CONTINUE
  160 CONTINUE
      RETURN
      END
      SUBROUTINE REDUCE (INFLTR,INSPACE,OUTFLTR,OTSPACE)
      INTEGER INSPACE,OTSPACE
      REAL            INFLTR(INSPACE),OUTFLTR(OTSPACE)
      INTEGER INHRMNC,OUTHRMC,SKIP,DOWN,OUTVCTR,INVCTR,START,STOP
      IF (INSPACE.NE.OTSPACE) GO TO 90
      DO 85 NVECTOR=1,OTSPACE
   85 OUTFLTR(NVECTOR)=INFLTR(NVECTOR)
      GO TO 125
   90 CONTINUE
      INHRMNC= INSPACE**.5    $  OUTHRMC=OTSPACE**.5
      SKIP=(INHRMNC-OUTHRMC)*2
      DOWN=OUTHRMC/2+1
      OUTFLTR(1)=INFLTR(1)
      IF (OTSPACE.EQ.1) RETURN
      DO 100 OUTVCTR=2,OUTHRMC
  100 OUTFLTR(OUTVCTR)=INFLTR(OUTVCTR)
      INVCTR=START=OUTHRMC+1
      DO 120 NROW=2,DOWN
      INVCTR=INVCTR+SKIP
      STOP=START+OUTHRMC*2-1
      DO 110 OUTVCTR=START,STOP
      OUTFLTR(OUTVCTR)=INFLTR(INVCTR)
      INVCTR=INVCTR+1
  110 CONTINUE
      START=STOP+1
  120 CONTINUE
  125 CONTINUE
      DO 130 I=1,4
  130 OUTFLTR(OTSPACE+I)=INFLTR(INSPACE+I)
      RETURN
      END
      SUBROUTINE SIZEIT(LTRSIZE,WIDTH,HEIGHT)
      INTEGER LTRSIZE,WIDTH,HEIGHT
```

```
      INTEGER LTRSIZE,WIDTH,HEIGHT
      COMMON/LETTERS/LETTER,NUMLTRS,RECNUM
      INTEGER LETTER(32,32),NUMLTRS,RECNUM
      INTEGER NCOL,NROW,STRTCOL,TEMP,STRTROW
      DO 100 NCOL=1,32
      DO 100 NROW=1,32
      IF (LETTER(NROW,NCOL).EQ.1) GO TO 110
  100 CONTINUE
  110 STRTCOL=NCOL
      DO 120 NCOL=1,32
      TEMP=33-NCOL
      DO 120 NROW=1,32
      IF (LETTER(NROW, TEMP).EQ.1) GO TO 130
  120 CONTINUE
  130 WIDTH=TEMP-STRTCOL+1
      IF (WIDTH.GE.1)  GO TO 135
      WIDTH=HEIGHT=0
      RETURN
  135 CONTINUE
      DO 140 NROW=1,32
      DO 140 NCOL=1,LTRSIZE
      IF(LETTER(NROW,NCOL).EQ.1) GO TO 150
  140 CONTINUE
  150 STRTROW=NROW
      DO 160 NROW=1,LTRSIZE
      TEMP=LTRSIZE+1-NROW
      DO 160 NCOL=1,LTRSIZE
      IF (LETTER(TEMP,NCOL).EQ.1) GO TO 170
  160 CONTINUE
  170 HEIGHT=TEMP-STRTROW+1
      RETURN
      END
      SUBROUTINE SORTIT(STRTCOL,STOPCOL,NSKIPS,STRTROW,STOPROW)
      INTEGER STRTCOL,STOPCOL,NSKIPS,STRTROW,STOPROW
      COMMON/RESULTS/RESULTS
      INTEGER RESULTS(52,133)
      INTEGER FLAG,XFRMCOL ,TEMP,LASTROW,DXMASK
      DXMASK=SHIFT(MASK(22),22)
      LASTROW=STOPROW-1
      DO 120 XFRMCOL=STRTCOL,STOPCOL,NSKIPS
  100 FLAG=0
      DO 110 NROW=STRTROW,LASTROW
      IF ((DXMASK.AND.RESULTS(NROW,XFRMCOL)).LE.(DXMASK.AND.RESULTS(NROW
     %+1,XFRMCOL))) GO TO 110
      TEMP=RESULTS(NROW,XFRMCOL)
      RESULTS(NROW,XFRMCOL)=RESULTS(NROW+1,XFRMCOL)
      RESULTS(NROW+1,XFRMCOL)=TEMP
      FLAG=1
  110 CONTINUE
      IF (FLAG.EQ.1) GO TO 100
  120 CONTINUE
      RETURN
      END
      SUBROUTINE XFORMIT(LTRSIZE,WIDTH,HEIGHT,ADJUST,WRATIO,HRATIO,
     %NSPACE,NUMROWS,NUMCOLS,ALPHNUM,LTRNUM,VEXPAND,XFORM,IMAGE,INVERSE,
     %SENTNUM)
      INTEGER LTRSIZE,WIDTH,HEIGHT,NSPACE,NUMROWS,NUMCOLS,ALPHNUM,LTRNUM
     %,SENTNUM
      LOGICAL ADJUST   $ REAL WRATIO,HRATIO
      LOGICAL VEXPAND,XFORM,IMAGE,INVERSE
      INTEGER LETTER(32,32),NUMLTRS,RECNUM
      COMMON/LETTERS/LETTER,NUMLTRS,RECNUM
      INTEGER RECFLTR
      REAL FLTRLTR(180),PROTO(180)
      COMMON/PROFLTR/FLTRLTR,PROTO,RECFLTR
      COMMON/XFORMS/FORTLTR
```

```
      COMPLEX FORTLTR(64,64)
      COMPLEX WORKER(64)
      INTEGER POWERS(7)
      INTEGER NN(2),NUMCOLS,NUMROWS,NCOL,NROW,FLAG,TOP,BOTTOM,ITEMP,
     %              COL,ROW,     LEFT,RIGHT,IFORM,ISIGN
      DIMENSION TRPLTR(32,32)
      DATA POWERS/2,4,8,16,32,64,128/
      FLAG=0
  100 CONTINUE
      CALL SIZEIT(LTRSIZE,WIDTH,HEIGHT)
      CALL TRPMLT(LETTER,TRPLTR,SENTNUM)
      NUMROWS=NUMCOLS=32
      NRINGS=NSPACE**.5
      IF(NUMROWS.LT.NRINGS) NUMROWS=NRINGS
      IF(NUMCOLS.LT.NRINGS) NUMCOLS=NRINGS
      IF (MOD(NUMROWS,2).EQ.1) NUMROWS=NUMROWS+1
      IF (MOD(NUMCOLS,2).EQ.1) NUMCOLS=NUMCOLS+1
      TOP =IABS(NUMROWS-LTRSIZE)/2
      IF (NUMROWS.LT.LTRSIZE) GO TO 150
      BOTTOM=TOP+1+LTRSIZE
      ITEMP=TOP
      DO 140 NCOL=1,LTRSIZE
      TOP=ITEMP
      DO 110 NROW=1,TOP
  110 FORTLTR(NROW,NCOL)=0
      DO 120 NROW=1,LTRSIZE
      TOP=TOP+1
  120 FORTLTR(TOP,NCOL)=TRPLTR(NROW,NCOL)
      DO 130 NROW=BOTTOM,NUMROWS
  130   FORTLTR(NROW,NCOL)=0
  140 CONTINUE
      GO TO 170
  150 CONTINUE
      TOP=TOP+1
      BOTTOM=TOP+NUMROWS-1
      DO 160 NCOL=1,LTRSIZE
      ROW=0
      DO 160 NROW=TOP,BOTTOM
      ROW=ROW+1
  160 FORTLTR(ROW,NCOL)=TRPLTR(NROW,NCOL)
  170 CONTINUE
      LEFT=IABS(NUMCOLS-LTRSIZE)/2
      IF(NUMCOLS.LT.LTRSIZE) GO TO 220
      RIGHT=LEFT+LTRSIZE+1
      DO 210 NROW=1,NUMROWS
      DO 180 NCOL=RIGHT,NUMCOLS
  180 FORTLTR(NROW,NCOL)=0
      ITEMP=RIGHT
      DO 190 NCOL=1,LTRSIZE
      ITEMP=ITEMP-1
      COL=LTRSIZE-NCOL+1
  190 FORTLTR(NROW,ITEMP)=FORTLTR(NROW,COL)
      DO 200 NCOL=1,LEFT
  200 FORTLTR(NROW,NCOL)=0
  210 CONTINUE
      GO TO 232
  220 CONTINUE
      LEFT=LEFT+1
      RIGHT=LEFT+NUMCOLS-1
      DO 230 NROW=1,NUMROWS
      COL=0
      DO 230 NCOL=LEFT,RIGHT
      COL=COL+1
  230 FORTLTR(NROW,COL)=FORTLTR(NROW,NCOL)
  232 CONTINUE
      CALL ARNGCPX(FORTLTR,64,64,FORTLTR,NUMROWS,NUMCOLS)
  235 CONTINUE
```

```
      IF (FLAG.GT.0) GO TO 245
      NN(1)=NUMROWS $ NN(2)=NUMCOLS  $NDIM=2 $ ISIGN=-1 $ IFORM=0
      COL=ROW=0
      DO 240 ITEMP=1,7
      IF (NUMROWS.EQ.POWERS(ITEMP))  ROW=1
240   IF(NUMCOLS.EQ.POWERS(ITEMP))  COL=1
245   CONTINUE
      IF ( ROW.EQ.1.AND. COL.EQ.1) GO TO 250
      CALL FOURT (FORTLTR,NN,NDIM,ISIGN,IFORM,WORKER)
      GO TO 255
250   CALL FOURT (FORTLTR,NN,NDIM,ISIGN,IFORM,0.0)
C
C     THIS IS THE PLACE TO PUT IN AN INVERSE PRINT ROUTINE THAT
C     FILTERS THE FOURIER TRANSFORM FROM THE MIDDLE AND THEN CALL SLPRT
C
255   CONTINUE
      IF (FLAG.GT.0) RETURN
      FLAG=FLAG+1
      RETURN
      END
      SUBROUTINE FOURT(DATA,NN,NDIM,ISIGN,IFORM,WORK)
C
C     THE COOLEY-TUKEY FAST FOURIER TRANSFORM IN USASI BASIC FORTRAN
C
C     TRANSFORM(K1,K2,...) = SUM(DATA(J1,J2,...)*EXP(ISIGN*2*PI*SQRT(-1)
C     *((J1-1)*(K1-1)/NN(1)+(J2-1)*(K2-1)/NN(2)+...))), SUMMED FOR ALL
C     J1, K1 BETWEEN 1 AND NN(1), J2, K2 BETWEEN 1 AND NN(2), ETC.
C     THERE IS NO LIMIT TO THE NUMBER OF SUBSCRIPTS.  DATA IS A
C     MULTIDIMENSIONAL COMPLEX ARRAY WHOSE REAL AND IMAGINARY
C     PARTS ARE ADJACENT IN STORAGE, SUCH AS FORTRAN IV PLACES THEM.
C     IF ALL IMAGINARY PARTS ARE ZERO (DATA ARE DISGUISED REAL), SET
C     IFORM TO ZERO TO CUT THE RUNNING TIME BY UP TO FORTY PERCENT.
C     OTHERWISE, IFORM = +1.  THE LENGTHS OF ALL DIMENSIONS ARE
C     STORED IN ARRAY NN, OF LENGTH NDIM.  THEY MAY BE ANY POSITIVE
C     INTEGERS, THO THE PROGRAM RUNS FASTER ON COMPOSITE INTEGERS, AND
C     ESPECIALLY FAST ON NUMBERS RICH IN FACTORS OF TWO.  ISIGN IS +1
C     OR -1.  IF A -1 TRANSFORM IS FOLLOWED BY A +1 ONE (OR A +1
C     BY A -1) THE ORIGINAL DATA REAPPEAR, MULTIPLIED BY NTOT (=NN(1)*
C     NN(2)*...).  TRANSFORM VALUES ARE ALWAYS COMPLEX, AND ARE RETURNED
C     IN ARRAY DATA, REPLACING THE INPUT.  IN ADDITION, IF ALL
C     DIMENSIONS ARE NOT POWERS OF TWO, ARRAY WORK MUST BE SUPPLIED,
C     COMPLEX OF LENGTH EQUAL TO THE LARGEST NON 2**K DIMENSION.
C     OTHERWISE, REPLACE WORK BY ZERO IN THE CALLING SEQUENCE.
C     NORMAL FORTRAN DATA ORDERING IS EXPECTED, FIRST SUBSCRIPT VARYING
C     FASTEST.  ALL SUBSCRIPTS BEGIN AT ONE.
C
C     RUNNING TIME IS MUCH SHORTER THAN THE NAIVE NTOT**2, BEING
C     GIVEN BY THE FOLLOWING FORMULA.  DECOMPOSE NTOT INTO
C     2**K2 * 3**K3 * 5**K5 * ....  LET SUM2 = 2*K2, SUMF = 3*K3 + 5*K5
C     + ... AND NF = K3 + K5 + ....  THE TIME TAKEN BY A MULTI-
C     DIMENSIONAL TRANSFORM ON THESE NTOT DATA IS T = T0 + NTOT*(T1+
C     T2*SUM2+T3*SUMF+T4*NF).  ON THE CDC 3300 (FLOATING POINT ADD TIME
C     OF SIX MICROSECONDS), T = 3000 + NTOT*(500+43*SUM2+68*SUMF+
C     320*NF) MICROSECONDS ON COMPLEX DATA.  IN ADDITION, THE
C     ACCURACY IS GREATLY IMPROVED, AS THE RMS RELATIVE ERROR IS
C     BOUNDED BY 3*2**(-B)*SUM(FACTOR(J)**1.5), WHERE B IS THE NUMBER
C     OF BITS IN THE FLOATING POINT FRACTION AND FACTOR(J) ARE THE
C     PRIME FACTORS OF NTOT.
C
C     PROGRAM BY NORMAN BRENNER FROM THE BASIC PROGRAM BY CHARLES
C     RADER.  RALPH ALTER SUGGESTED THE IDEA FOR THE DIGIT REVERSAL.
C     MIT LINCOLN LABORATORY, AUGUST 1967.  THIS IS THE FASTEST AND MOST
C     VERSATILE VERSION OF THE FFT KNOWN TO THE AUTHOR.  SHORTER PRO-
C     GRAMS FOUR1 AND FOUR2 RESTRICT DIMENSION LENGTHS TO POWERS OF TWO.
C     SEE-- IEEE AUDIO TRANSACTIONS (JUNE 1967), SPECIAL ISSUE ON FFT.
```

```
C
C     THE DISCRETE FOURIER TRANSFORM PLACES THREE RESTRICTIONS UPON THE
C     DATA.
C     1.  THE NUMBER OF INPUT DATA AND THE NUMBER OF TRANSFORM VALUES
C     MUST BE THE SAME.
C     2.  BOTH THE INPUT DATA AND THE TRANSFORM VALUES MUST REPRESENT
C     EQUISPACED POINTS IN THEIR RESPECTIVE DOMAINS OF TIME AND
C     FREQUENCY.  CALLING THESE SPACINGS DELTAT AND DELTAF, IT MUST BE
C     TRUE THAT DELTAF=2*PI/(NN(I)*DELTAT).  OF COURSE, DELTAT NEED NOT
C     BE THE SAME FOR EVERY DIMENSION.
C     3.  CONCEPTUALLY AT LEAST, THE INPUT DATA AND THE TRANSFORM OUTPUT
C     REPRESENT SINGLE CYCLES OF PERIODIC FUNCTIONS.
C
C     EXAMPLE 1.  THREE-DIMENSIONAL FORWARD FOURIER TRANSFORM OF A
C     COMPLEX ARRAY DIMENSIONED 32 BY 25 BY 13 IN FORTRAN IV.
C     DIMENSION DATA(32,25,13),WORK(50),NN(3)
C     COMPLEX DATA
C     DATA NN/32,25,13/
C     DO 1 I=1,32
C     DO 1 J=1,25
C     DO 1 K=1,13
C  1  DATA(I,J,K)=COMPLEX VALUE
C     CALL FOURT(DATA,NN,3,-1,1,WORK)
C
C     EXAMPLE 2.  ONE-DIMENSIONAL FORWARD TRANSFORM OF A REAL ARRAY OF
C     LENGTH 64 IN FORTRAN II.
C     DIMENSION DATA(2,64)
C     DO 2 I=1,64
C     DATA(1,I)=REAL PART
C  2  DATA(2,I)=0.
C     CALL FOURT(DATA,64,1,-1,0,0)
C
      DIMENSION DATA(1),NN(1),IFACT(32),WORK(1)
      TWOPI=6.283185307
      WR=0.
      WI=0.
      WSTPI=0.
      WSTPR=0.
      IF(NDIM-1)920,1,1
    1 NTOT=2
      DO 2 IDIM=1,NDIM
      IF(NN(IDIM))920,920,2
    2 NTOT=NTOT*NN(IDIM)
C
C     MAIN LOOP FOR EACH DIMENSION
C
      NP1=2
      DO 910 IDIM=1,NDIM
      N=NN(IDIM)
      NP2=NP1*N
      IF(N-1)920,900,5
C
C     FACTOR N
C
    5 M=N
      NTWO=NP1
      IF=1
      IDIV=2
   10 IQUOT=M/IDIV
      IREM=M-IDIV*IQUOT
      IF(IQUOT-IDIV)50,11,11
   11 IF(IREM)20,12,20
   12 NTWO=NTWO+NTWO
      M=IQUOT
      GO TO 10
   20 IDIV=3
      IQUOT=M/IDIV
```

```
      IQUOT=M/IDIV
      IREM=M-IDIV*IQUOT
      IF(IQUOT-IDIV)60,31,31
   31 IF(IREM)40,32,40
   32 IFACT(IF)=IDIV
      IF=IF+1
      M=IQUOT
      GO TO 30
   40 IDIV=IDIV+2
      GO TO 30
   50 IF(IREM)60,51,60
   51 NTWO=NTWO+NTWO
      GO TO 70
   60 IFACT(IF)=M
C
C     SEPARATE FOUR CASES--
C        1. COMPLEX TRANSFORM OR REAL TRANSFORM FOR THE 4TH, 5TH, ETC.
C           DIMENSIONS.
C        2. REAL TRANSFORM FOR THE 2ND OR 3RD DIMENSION.  METHOD--
C           TRANSFORM HALF THE DATA, SUPPLYING THE OTHER HALF BY CON-
C           JUGATE SYMMETRY.
C        3. REAL TRANSFORM FOR THE 1ST DIMENSION, N ODD.  METHOD--
C           TRANSFORM HALF THE DATA AT EACH STAGE, SUPPLYING THE OTHER
C           HALF BY CONJUGATE SYMMETRY.
C        4. REAL TRANSFORM FOR THE 1ST DIMENSION, N EVEN.  METHOD--
C           TRANSFORM A COMPLEX ARRAY OF LENGTH N/2 WHOSE REAL PARTS
C           ARE THE EVEN NUMBERED REAL VALUES AND WHOSE IMAGINARY PARTS
C           ARE THE ODD NUMBERED REAL VALUES.  SEPARATE AND SUPPLY
C           THE SECOND HALF BY CONJUGATE SYMMETRY.
C
   70 NON2=NP1*(NP2/NTWO)
      ICASE=1
      IF(IDIM-4)71,90,90
   71 IF(IFORM)72,72,90
   72 ICASE=2
      IF(IDIM-1)73,73,90
   73 ICASE=3
      IF(NTWO-NP1)90,90,74
   74 ICASE=4
      NTWO=NTWO/2
      N=N/2
      NP2=NP2/2
      NTOT=NTOT/2
      I=3
      DO 80 J=2,NTOT
      DATA(J)=DATA(I)
   80 I=I+2
   90 I1RNG=NP1
      IF(ICASE-2)100,95,100
   95 I1RNG=NP0*(1+NPREV/2)
C
C     SHUFFLE ON THE FACTORS OF TWO IN N.  AS THE SHUFFLING
C     CAN BE DONE BY SIMPLE INTERCHANGE, NO WORKING ARRAY IS NEEDED
C
  100 IF(NTWO-NP1)600,600,110
  110 NP2HF=NP2/2
      J=1
      DO 150 I2=1,NP2,NON2
      IF(J-I2)120,130,130
  120 I1MAX=I2+NON2-2
      DO 125 I1=I2,I1MAX,2
      DO 125 I3=I1,NTOT,NP2
      J3=J+I3-I2
      TEMPR=DATA(I3)
      TEMPI=DATA(I3+1)
      DATA(I3)=DATA(J3)
      DATA(I3+1)=DATA(J3+1)
```

```
      DATA(J3)=TEMPR
  125 DATA(J3+1)=TEMPI
  130 M=NP2HF
  140 IF(J-M)150,150,145
  145 J=J-M
      M=M/2
      IF(M-NON2)150,140,140
  150 J=J+M
C
C     MAIN LOOP FOR FACTORS OF TWO.  PERFORM FOURIER TRANSFORMS OF
C     LENGTH FOUR, WITH ONE OF LENGTH TWO IF NEEDED.  THE TWIDDLE FACTOR
C     W=EXP(ISIGN*2*PI*SQRT(-1)*M/(4*MMAX)).  CHECK FOR W=ISIGN*SQRT(-1)
C     AND REPEAT FOR W=ISIGN*SQRT(-1)*CONJUGATE(W).
C
      NON2T=NON2+NON2
      IPAR=NTWO/NP1
  310 IF(IPAR-2)350,330,320
  320 IPAR=IPAR/4
      GO TO 310
  330 DO 340 I1=1,I1RNG,2
      DO 340 J3=I1,NON2,NP1
      DO 340 K1=J3,NTOT,NON2T
      K2=K1+NON2
      TEMPR=DATA(K2)
      TEMPI=DATA(K2+1)
      DATA(K2)=DATA(K1)-TEMPR
      DATA(K2+1)=DATA(K1+1)-TEMPI
      DATA(K1)=DATA(K1)+TEMPR
  340 DATA(K1+1)=DATA(K1+1)+TEMPI
  350 MMAX=NON2
  360 IF(MMAX-NP2HF)370,600,600
  370 LMAX=MAX0(NON2T,MMAX/2)
      IF(MMAX-NON2)405,405,380
  380 THETA=-TWOPI*FLOAT(NON2)/FLOAT(4*MMAX)
      IF(ISIGN)400,390,390
  390 THETA=-THETA
  400 WR=COS(THETA)
      WI=SIN(THETA)
      WSTPR=-2.*WI*WI
      WSTPI=2.*WR*WI
  405 DO 570 L=NON2,LMAX,NON2T
      M=L
      IF(MMAX-NON2)420,420,410
  410 W2R=WR*WR-WI*WI
      W2I=2.*WR*WI
      W3R=W2R*WR-W2I*WI
      W3I=W2R*WI+W2I*WR
  420 DO 530 I1=1,I1RNG,2
      DO 530 J3=I1,NON2,NP1
      KMIN=J3+IPAR*M
      IF(MMAX-NON2)430,430,440
  430 KMIN=J3
  440 KDIF=IPAR*MMAX
  450 KSTEP=4*KDIF
      DO 520 K1=KMIN,NTOT,KSTEP
      K2=K1+KDIF
      K3=K2+KDIF
      K4=K3+KDIF
      IF(MMAX-NON2)460,460,480
  460 U1R=DATA(K1)+DATA(K2)
      U1I=DATA(K1+1)+DATA(K2+1)
      U2R=DATA(K3)+DATA(K4)
      U2I=DATA(K3+1)+DATA(K4+1)
      U3R=DATA(K1)-DATA(K2)
      U3I=DATA(K1+1)-DATA(K2+1)
      IF(ISIGN)470,475,475
```

```
470   U4R=DATA(K3+1)-DATA(K4+1)
      U4I=DATA(K4)-DATA(K3)
      GO TO 510
475   U4R=DATA(K4+1)-DATA(K3+1)
      U4I=DATA(K3)-DATA(K4)
      GO TO 510
480   T2R=W2R*DATA(K2)-W2I*DATA(K2+1)
      T2I=W2R*DATA(K2+1)+W2I*DATA(K2)
      T3R=WR*DATA(K3)-WI*DATA(K3+1)
      T3I=WR*DATA(K3+1)+WI*DATA(K3)
      T4R=W3R*DATA(K4)-W3I*DATA(K4+1)
      T4I=W3R*DATA(K4+1)+W3I*DATA(K4)
      U1R=DATA(K1)+T2R
      U1I=DATA(K1+1)+T2I
      U2R=T3R+T4R
      U2I=T3I+T4I
      U3R=DATA(K1)-T2R
      U3I=DATA(K1+1)-T2I
      IF(ISIGN)490,500,500
490   U4R=T3I-T4I
      U4I=T4R-T3R
      GO TO 510
500   U4R=T4I-T3I
      U4I=T3R-T4R
510   DATA(K1)=U1R+U2R
      DATA(K1+1)=U1I+U2I
      DATA(K2)=U3R+U4R
      DATA(K2+1)=U3I+U4I
      DATA(K3)=U1R-U2R
      DATA(K3+1)=U1I-U2I
      DATA(K4)=U3R-U4R
520   DATA(K4+1)=U3I-U4I
      KMIN=4*(KMIN-J3)+J3
      KDIF=KSTEP
      IF(KDIF-NP2)450,530,530
530   CONTINUE
      M=MMAX-M
      IF(ISIGN)540,550,550
540   TEMPR=WR
      WR=-WI
      WI=-TEMPR
      GO TO 550
550   TEMPR=WR
      WR=WI
      WI=TEMPR
560   IF(M-LMAX)565,565,410
565   TEMPR=WR
      WR=WR*WSTPR-WI*WSTPI+WR
570   WI=WI*WSTPR+TEMPR*WSTPI+WI
      IPAR=3-IPAR
      MMAX=MMAX+MMAX
      GO TO 350
C
C     MAIN LOOP FOR FACTORS NOT EQUAL TO TWO.  APPLY THE TWIDDLE FACTOR
C     W=EXP(ISIGN*2*PI*SQRT(-1)*(J2-1)*(J1-J2)/(NP2*IFP1)), THEN
C     PERFORM A FOURIER TRANSFORM OF LENGTH IFACT(IF), MAKING USE OF
C     CONJUGATE SYMMETRIES.
C
600   IF(NTWO-NP2)605,700,700
605   IFP1=NON2
      IF=1
      NP1HF=NP1/2
610   IFP2=IFP1/IFACT(IF)
      U1RNG=NP2
      IF(ICASE-3)612,611,612
```

```
      U2STP=NP2/IFACT(IF)
      U1RG2=(J2STP+IFP2)/2
  612 U2MIN=1+IFP2
      IF(IFP1-NP2)615,640,640
  615 DO 635 J2=J2MIN,IFP1,IFP2
      THETA=-TWOPI*FLOAT(J2-1)/FLOAT(NP2)
      IF(ISIGN)625,620,620
  620 THETA=-THETA
  625 SINTH=SIN(THETA/2.)
      WSTPR=-2.*SINTH*SINTH
      WSTPI=SIN(THETA)
      WR=WSTPR+1.
      WI=WSTPI
      U1MIN=J2+IFP1
      DO 635 J1=J1MIN,J1RNG,IFP1
      I1MAX=J1+I1RNG-2
      DO 630 I1=J1,I1MAX,2
      DO 630 I3=I1,NTOT,NP2
      U3MAX=I3+IFP2-NP1
      DO 630 J3=I3,J3MAX,NP1
      TEMPR=DATA(J3)
      DATA(J3)=DATA(J3)*WR-DATA(J3+1)*WI
  630 DATA(J3+1)=TEMPR*WI+DATA(J3+1)*WR
      TEMPR=WR
      WR=WR*WSTPR-WI*WSTPI+WR
  635 WI=TEMPR*WSTPI+WI*WSTPR+WI
  640 THETA=-TWOPI/FLOAT(IFACT(IF))
      IF(ISIGN)650,645,645
  645 THETA=-THETA
  650 SINTH=SIN(THETA/2.)
      WSTPR=-2.*SINTH*SINTH
      WSTPI=SIN(THETA)
      KSTEP=2*N/IFACT(IF)
      KRANG=KSTEP*(IFACT(IF)/2)+1
      DO 698 I1=1,I1RNG,2
      DO 698 I3=I1,NTOT,NP2
      DO 690 KMIN=1,KRANG,KSTEP
      U1MAX=I3+J1RNG-IFP1
      DO 680 J1=I3,J1MAX,IFP1
      U3MAX=J1+IFP2-NP1
      DO 680 J3=J1,J3MAX,NP1
      U2MAX=J3+IFP1-IFP2
      K=KMIN+(J3-J1+(J1-I3)/IFACT(IF))/NP1HF
      IF(KMIN-1)655,655,665
  655 SUMR=0.
      SUMI=0.
      DO 660 J2=J3,J2MAX,IFP2
      SUMR=SUMR+DATA(J2)
  660 SUMI=SUMI+DATA(J2+1)
      WORK(K)=SUMR
      WORK(K+1)=SUMI
      GO TO 680
  665 KCONJ=K+2*(N-KMIN+1)
      U2=J2MAX
      SUMR=DATA(J2)
      SUMI=DATA(J2+1)
      OLDSR=0.
      OLDSI=0.
      U2=J2-IFP2
  670 TEMPR=SUMR
      TEMPI=SUMI
      SUMR=TWOWR*SUMR-OLDSR+DATA(J2)
      SUMI=TWOWR*SUMI-OLDSI+DATA(J2+1)
      OLDSR=TEMPR
      OLDSI=TEMPI
```

```
      IF(J2-J3)675,675,670
  675 TEMPR=WR*SUMR-OLDSR+DATA(J2)
      TEMPI=WI*SUMI
      WORK(K)=TEMPR-TEMPI
      WORK(KCONJ)=TEMPR+TEMPI
      TEMPR=WR*SUMI-OLDSI+DATA(J2+1)
      TEMPI=WI*SUMR
      WORK(K+1)=TEMPR+TEMPI
      WORK(KCONJ+1)=TEMPR-TEMPI
  680 CONTINUE
      IF(KMIN-1)685,685,686
  685 WR=WSTPR+1.
      WI=WSTPI
      GO TO 690
  686 TEMPR=WR
      WR=WR*WSTPR-WI*WSTPI+WR
      WI=TEMPR*WSTPI+WI*WSTPR+WI
  690 TWOWR=WR+WR
      IF(ICASE-3)692,691,692
  691 IF(IFP1-NP2)695,692,692
  692 K=1
      I2MAX=I3+NP2-NP1
      DO 693 I2=I3,I2MAX,NP1
      DATA(I2)=WORK(K)
      DATA(I2+1)=WORK(K+1)
  693 K=K+2
      GO TO 698
C
C     COMPLETE A REAL TRANSFORM IN THE 1ST DIMENSION, N ODD, BY CON-
C     JUGATE SYMMETRIES AT EACH STAGE.
C
  695 J3MAX=I3+IFP2-NP1
      DO 697 J3=I3,J3MAX,NP1
      J2MAX=J3+NP2-J2STP
      DO 697 J2=J3,J2MAX,J2STP
      J1MAX=J2+J1RG2-IFP2
      J1CNJ=J3+J2MAX+J2STP-J2
      DO 697 J1=J2,J1MAX,IFP2
      K=1+J1-I3
      DATA(J1)=WORK(K)
      DATA(J1+1)=WORK(K+1)
      IF(J1-J2)697,697,696
  696 DATA(J1CNJ)=WORK(K)
      DATA(J1CNJ+1)=-WORK(K+1)
  697 J1CNJ=J1CNJ-IFP2
  698 CONTINUE
      IF=IF+1
      IFP1=IFP2
      IF(IFP1-NP1)700,700,610
C
C     COMPLETE A REAL TRANSFORM IN THE 1ST DIMENSION, N EVEN, BY CON-
C     JUGATE SYMMETRIES.
C
  700 GO TO (900,800,900,701),ICASE
  701 NHALF=N
      N=N+N
      THETA=-TWOPI/FLOAT(N)
      IF(ISIGN)703,702,702
  702 THETA=-THETA
  703 SINTH=SIN(THETA/2.)
      WSTPR=-2.*SINTH*SINTH
      WSTPI=SIN(THETA)
      WR=WSTPR+1.
      WI=WSTPI
      IMIN=3
      JMIN=2*NHALF+1
```

```
      JMIN=2*NHALF-1
      GO TO 725
  710 J=JMIN
      DO 720 I=IMIN,NTOT,NP2
      SUMR=(DATA(I)+DATA(J))/2.
      SUMI=(DATA(I+1)+DATA(J+1))/2.
      DIFR=(DATA(I)-DATA(J))/2.
      DIFI=(DATA(I+1)-DATA(J+1))/2.
      TEMPR=WR*SUMI+WI*DIFR
      TEMPI=WI*SUMI-WR*DIFR
      DATA(I)=SUMR+TEMPR
      DATA(I+1)=DIFI+TEMPI
      DATA(J)=SUMR-TEMPR
      DATA(J+1)=-DIFI+TEMPI
  720 J=J+NP2
      IMIN=IMIN+2
      JMIN=JMIN-2
      TEMPR=WR
      WR=WR*WSTPR-WI*WSTPI+WR
      WI=TEMPR*WSTPI+WI*WSTPR+WI
  725 IF(IMIN-JMIN)710,730,740
  730 IF(ISIGN)731,740,740
  731 DO 735 I=IMIN,NTOT,NP2
  735 DATA(I+1)=-DATA(I+1)
  740 NP2=NP2+NP2
      NTOT=NTOT+NTOT
      J=NTOT+1
      IMAX=NTOT/2+1
  745 IMIN=IMAX-2*NHALF
      I=IMIN
      GO TO 755
  750 DATA(J)=DATA(I)
      DATA(J+1)=-DATA(I+1)
  755 I=I+2
      J=J-2
      IF(I-IMAX)750,760,760
  760 DATA(J)=DATA(IMIN)-DATA(IMIN+1)
      DATA(J+1)=0.
      IF(I-J)770,780,780
  765 DATA(J)=DATA(I)
      DATA(J+1)=DATA(I+1)
  770 I=I-2
      J=J-2
      IF(I-IMIN)775,775,765
  775 DATA(J)=DATA(IMIN)+DATA(IMIN+1)
      DATA(J+1)=0.
      IMAX=IMIN
      GO TO 745
  780 DATA(1)=DATA(1)+DATA(2)
      DATA(2)=0.
      GO TO 900
C
C     COMPLETE A REAL TRANSFORM FOR THE 2ND OR 3RD DIMENSION BY
C     CONJUGATE SYMMETRIES.
C
  800 IF(I1RNG-NP1)805,900,900
  805 DO 860 I3=1,NTOT,NP2
      I2MAX=I3+NP2-NP1
      DO 860 I2=I3,I2MAX,NP1
      IMIN=I2+I1RNG
      IMAX=I2+NP1-2
      JMAX=2*I3+NP1-IMIN
      IF(I2-I3)820,820,810
  810 JMAX=JMAX+NP2
  820 IF(IDIM-2)850,850,830
  830 J=JMAX+NP0
      DO 840 I=IMIN,IMAX,2
```

```
      DO 840 I=IMIN,IMAX,NP0
      DATA(I)=DATA(J)
      DATA(I+1)=-DATA(J+1)
840   U=J-2
850   U=JMAX
      DO 860 I=IMIN,IMAX,NP0
      DATA(I)=DATA(J)
      DATA(I+1)=-DATA(J+1)
860   U=J-NP0
C
C     END OF LOOP ON EACH DIMENSION
C
900   NP0=NP1
      NP1=NP2
910   NPREV=N
920   RETURN
      END
      SUBROUTINE TRPMLT(LETTER,TRPLTR,SENTNUM)
C    THIS ROUTINE DOES TRAPEZOID BUILD AND MULTIPLY
      INTEGER LETTER(32,32),SENTNUM
      DIMENSION TRPLTR(32,32),TRPWDW(32)
    1 FORMAT(1H0)
    2 FORMAT(1H ,24("*"),/1H ,"* TRAPEZOID PARAMETERS *   THIS TRAPEZOID
     % WAS APPLIED TO WINDOWS OF SENTENCE NUMBER ",I2,/1H ,24("*"))
    3 FORMAT(1H ,/," LENGTH OF TRAPEZOID PLATEAU = ",I2)
    4 FORMAT(1H ,/," VARIANCE ON NORMALIZED WINDOW = ",F5.2)
    5 FORMAT(1H ,/," TRAPEZOID WINDOW ARRAY IS: ")
    6 FORMAT("       ",10F6.3)
    7 FORMAT(1H ,/," THIS IS A NORMAL WINDOW APPLIED TO SENTENCE NUMBER
     %",I2)
      IF (PSKIP.EQ."OK") GO TO 45
CDATA NORMALIZED WINDOW SET IPLEN=0 AND TRPHT=VARIANCE
      IPLEN = 0      $     TRPHT = 5.
C
C THIS PART BUILDS THE TRAPEZOID
C
      INCLN=(32-IPLEN)/2
      IF (IPLEN.NE.0) GO TO 9
      DO 8 I=1,16
    8 TRPWDW(17-I)=TRPWDW(16+I)=EXP(-((I-1)**2)/(2.*((TRPHT)**2)))
      WRITE 1
      WRITE 7,SENTNUM
      GO TO 44
    9 IF (INCLN.EQ.0) GO TO 35
      DO 10 I=1,INCLN
      TRPWDW(I)=TRPWDW(33-I)=(I/(INCLN+1.))*TRPHT
   10 CONTINUE
      DO 20 I=1,IPLEN
      TRPWDW(I+INCLN)=1.
   20 CONTINUE
      IF((32-(2*INCLN)).EQ.IPLEN) GO TO 43
      IODD=INCLN+1
      DO 30 I=1,IODD
      TRPWDW(33-I)=(I/(INCLN+2.))*TRPHT
   30 CONTINUE
      GO TO 43
   35 DO 40 I=1,16
   40 TRPWDW(I)=TRPWDW(33-I)=1.
   43 CONTINUE
      WRITE 1
      WRITE 1
      WRITE 2,SENTNUM
      WRITE 3,IPLEN
   44 WRITE 4,TRPHT
      WRITE 5
      WRITE 6,TRPWDW
C
```

```
C
C THIS PART DOES THE MULTIPLICATION
C
   45   DO 60 I=1,32
        DO 50 J=1,32
   50   TRPLTR(J,I)=TRPWDW(I)*(FLOAT(LETTER(J,I)))
   60   CONTINUE
        PSKIP="OK"
        RETURN
        END


**********          ROY00SV   //// END  OF LIST ////
```

Appendix F

Data Output For

Various Standard Deviations

RESULTS MATRIX SHOWING THE PROTOTYPE-TO-WINDOW
DISTANCES FOR PROTOTYPES WHOSE WIDTH VARIATION ENABLES
THE PROTOTYPE TO BE COMPARED TO WINDOWS OF WIDTH 32 COLUMNS
MATRIX IS SORTED BY DISTANCE WITHIN THE SENTENCE
COLUMNS. THE 1 TO 13 CLOSEST IDENTIFYING PROTOTYPES ARE
SHOWN FOR EACH WINDOW OF WIDTH 32 BEGINNING AND ENDING AT
THE SENTENCE COLUMN INDICATED BY THE LEFTMOST MATRIX COLUMN.

PROTOTYPES IN THE MATRIX ARE: A1 B1 C1 D1 E1 F1 G1 H1 I1 J1 K1 L1 M1 N1 O1 P1 Q

Window / Prototype Standard Deviations = 2 col

MATRIX 1 OF 1

MATRIX OF THE 1 TO 5 CLOSEST IDENTIFYING PROTOTYPES FOR WINDOWS OF WIDTH 32 S

SENTENCE COLUMNS	*	GUESS 1 PROTO/DX	GUESS 2 PROTO/DX	GUESS 3 PROTO/DX	GUESS 4 PROTO/DX	GUESS 5 PROTO/DX
1- 32	*	V1- .439	M1- .736	C1- .944	Y1- .953	I1-1.004
2- 33	*	V1- .619	C1- .847	O1- .907	M1- .955	U1- .989
3- 34	*	J1- .729	W1- .859	O1- .874	B1- .886	O1- .952
4- 35	*	J1- .340	M1- .836	A1- .869	T1- .953	K1- .859
5- 36	*	J1- .330	T1- .738	K1- .701	A1- .717	T1- .731
6- 37	*	T1- .292	I1- .449	J1- .593	A1- .650	K1- .662
7- 38	*	T1- .163	I1- .219	Y1- .665	A1- .720	K1- .756
8- 39	*	I1- .351	T1- .453	Y1- .701	F1- .808	A1- .853
9- 40	*	I1- .519	F1- .681	T1- .726	Y1- .807	O1- .900
10- 41	*	F1- .535	B1- .784	B1- .795	S1- .849	O1- .850
11- 42	*	F1- .423	B1- .464	B1- .475	E1- .617	H1- .657
12- 43	*	B1- .000	E1- .413	B1- .431	R1- .556	F1- .577
13- 44	*	B1- .374	E1- .534	B1- .563	B1- .671	O1- .675
14- 45	*	B1- .560	J1- .669	E1- .698	E1- .715	T1- .731
15- 46	*	T1- .630	J1- .669	K1- .693	A1- .713	I1- .728
16- 47	*	T1- .455	I1- .587	K1- .664	A1- .719	Y1- .789
17- 48	*	T1- .456	I1- .502	Y1- .739	K1- .775	A1- .794
18- 49	*	T1- .615	I1- .631	Y1- .751	M1- .758	V1- .879
19- 50	*	T1- .778	M1- .794	Y1- .809	I1- .813	Y1- .874
20- 51	*	X1- .710	Y1- .821	T1- .833	N1- .851	A1- .892
21- 52	*	X1- .640	T1- .719	K1- .723	T1- .762	Y1- .782
22- 53	*	K1- .589	T1- .643	T1- .682	X1- .734	Y1- .735
23- 54	*	T1- .593	K1- .656	I1- .677	Y1- .745	T1- .800
24- 55	*	I1- .685	T1- .691	M1- .796	Y1- .821	K1- .851
25- 56	*	I1- .790	T1- .799	M1- .896	V1- .903	Y1- .926
26- 57	*	O1- .723	I1- .908	V1- .909	T1- .944	T1- .994
27- 58	*	O1- .360	G1- .779	O1- .847	V1- .853	I1- .920
28- 59	*	O1- .088	G1- .638	O1- .689	J1- .796	O1- .864
29- 60	*	O1- .347	O1- .654	U1- .824	G1- .832	O1- .729
30- 61	*	O1- .527	O1- .620	O1- .736	U1- .717	O1- .723
31- 62	*	O1- .508	O1- .731	J1- .807	N1- .838	O1- .916
32- 63	*	O1- .634	J1- .736	V1- .828	O1- .905	T1- .976
33- 64	*	J1- .800	O1- .816	I1- .822	N1- .851	T1- .927
34- 65	*	I1- .724	T1- .870	G1- .929	A1- .935	O1- .947
35- 66	*	I1- .797	G1- .819	O1- .845	O1- .887	T1- .918
36- 67	*	O1- .761	G1- .788	O1- .818	O1- .828	I1- .939
37- 68	*	J1- .677	O1- .784	O1- .853	B1- .886	G1- .929
38- 69	*	J1- .358	T1- .822	K1- .879	A1- .879	O1- .883
39- 70	*	J1- .705	T1- .866	K1- .788	I1- .717	A1- .767
40- 71	*	T1- .219	I1- .426	J1- .822	K1- .687	A1- .896
41- 72	*	T1- .191	T1- .216	Y1- .781	A1- .784	K1- .790
42- 73	*	I1- .361	T1- .428	Y1- .779	F1- .780	A1- .891
43- 74	*	I1- .642	F1- .782	T1- .813	O1- .873	Y1- .922
44- 75	*	F1- .505	O1- .794	B1- .869	I1- .914	G1- .915

```
 46- 77   * O1- .438  O1- .[illegible]  G1- .683  L1- .737  C1- .805
 47- 78   * O1- .000  O1- .832  C1- .767  B1- .852  O1- .924
 48- 79   * O1- .319  O1- .717  C1- .863  J1- .902  C1- .908
 49- 80   * O1- .521  M1- .771  C1- .846  G1- .921  C1- .986
 50- 81   * J1- .622  T1- .918  O1- .919  A1- .938  T1- .953
 51- 82   * J1- .869  T1- .744  A1- .809  I1- .928  K1- .976
 52- 83   * T1- .576  J1- .849  I1- .896  A1- .734  K1- .913
 53- 84   * T1- .569  I1- .620  A1- .751  Y1- .813  Y1- .857
 54- 85   * T1- .368  I1- .670  X1- .679  Y1- .739  A1- .793
 55- 86   * X1- .628  T1- .650  A1- .717  Y1- .721  K1- .725
 56- 87   * K1- .604  T1- .631  A1- .699  A1- .707  Y1- .734
 57- 88   * T1- .484  K1- .564  A1- .638  I1- .649  J1- .655
 58- 89   * T1- .274  I1- .432  K1- .639  A1- .651  Y1- .670
 59- 90   * T1- .244  I1- .256  Y1- .648  A1- .754  K1- .780
 60- 91   * I1- .387  T1- .[illegible]  Y1- .709  F1- .738  V1- .891
 61- 92   * I1- .567  S1- .691  T1- .703  Y1- .830  C1- .865
 62- 93   * F1- .513  B1- .737  B1- .829  G1- .825  S1- .891
 63- 94   * S1- .423  P1- .[illegible]  S1- .525  E1- .527  K1- .630
 64- 95   * E1- .661  B1- .[illegible]  T1- .[illegible]  W1- .641  C1- .668
 65- 96   * E1- .390  B1- .570  S1- .607  A1- .657  F1- .682
 66- 97   * S1- .[illegible]  B1- .679  S1- .741  T1- .713  K1- .724
 67- 98   * B1- .771  I1- .796  C1- .810  T1- .822  T1- .825
 68- 99   * O1- .714  T1- .811  C1- .[illegible]  B1- .851  T1- .877
 69-100   * O1- .587  O1- .736  C1- .706  G1- .775  C1- .856
 70-101   * O1- .[illegible]  O1- .689  C1- .702  G1- .711  C1- .869
 71-102   * J1- .520  O1- .[illegible]  C1- .799  G1- .814  C1- .871
 72-103   * J1- .307  T1- .719  A1- .814  I1- .815  K1- .818
 73-104   * J1- .464  T1- .466  B1- .586  K1- .716  A1- .717
 74-105   * T1- .233  I1- .295  Y1- .706  A1- .724  J1- .746
 75-106   * I1- .267  T1- .384  X1- .692  F1- .801  A1- .836
 76-107   * I1- .536  F1- .[illegible]  T1- .677  Y1- .732  S1- .840
 77-108   * F1- .[illegible]  P1- .[illegible]  S1- .731  B1- .741  T1- .765
 78-109   * F1- .[illegible]  P1- .612  C1- .577  S1- .[illegible]  S1- .633
 79-110   * F1- .380  P1- .[illegible]  S1- .572  R1- .[illegible]  B1- .525
 80-111   * F1- .572  P1- .727  S1- .738  R1- .755  P1- .807
 81-112   * F1- .870  J1- .879  P1- .879  W1- .881  E1- .896
 82-113   * A1- .789  R1- .912  E1- .947  J1- .948  I1- .967
 83-114   * A1- .[illegible]  J1- .[illegible]  B1- .[illegible]  I1- .940  T1- .943
 84-115   * A1- .597  T1- .724  J1- .738  K1- .824  T1- .830
 85-116   * T1- .510  T1- .[illegible]  K1- .734  A1- .774  T1- .797
 86-117   * T1- .[illegible]  I1- .[illegible]  Y1- .[illegible]  K1- .820  A1- .[illegible]
 87-118   * T1- .610  T1- .651  Y1- .926  A1- .933  V1- .973
 88-119   * I1- .77[illegible]  T1- .910  C1- .945  V1- .[illegible]  A1-1.008
 89-120   * C1- .767  I1- .[illegible]  C1- .908  G1- .[illegible]  T1- .968
 90-121   * C1- .638  G1- .[illegible]  C1- .714  O1- .[illegible]  C1- .[illegible]
 91-122   * G1- .775  O1- .[illegible]  C1- .[illegible]  O1- .732  I1- .876
 92-123   * G1- .[illegible]  O1- .[illegible]  C1- .638  I1- .[illegible]  C1- .767
 93-124   * G1- .[illegible]  O1- .[illegible]  I1- .732  L1- .[illegible]  C1- .804
 94-125   * G1- .971  S1- .877  C1- .897  O1- .847  O1- .932
 95-126   * K1- .732  J1- .[illegible]  Y1- .927  T1- .[illegible]  F1- .966
 96-127   * K1- .[illegible]  J1- .[illegible]  T1- .735  Y1- .821  T1- .926
 97-128   * T1- .[illegible]  K1- .[illegible]  J1- .[illegible]  I1- .[illegible]  Y1- .775
 98-129   * T1- .[illegible]  I1- .[illegible]  K1- .715  Y1- .[illegible]  V1- .812
 99-130   * I1- .[illegible]  T1- .[illegible]  K1- .710  V1- .732  Y1- .863
100-131   * I1- .733  M1- .799  T1- .811  V1- .[illegible]  Y1- .962
101-132   * M1- .923  V1- .[illegible]  I1- .[illegible]  F1- .[illegible]  T1-1.021
102-133   * F1- .[illegible]  M1-1.[illegible]  V1-1.[illegible]  B1-1.[illegible]  T1-1.115
```

RESULTS MATRIX SHOWING THE PROTOTYPE-TO-WINDOW
DISTANCES FOR PROTOTYPES WHOSE WIDTH VARIATION ENABLES
THE PROTOTYPE TO BE COMPARED TO WINDOWS OF WIDTH 32 COLUMNS
MATRIX IS SORTED BY DISTANCE WITHIN THE SENTENCE
COLUMNS. THE 1 TO 13 CLOSEST IDENTIFYING PROTOTYPES ARE
SHOWN FOR EACH WINDOW OF WIDTH 32 BEGINNING AND ENDING AT
THE SENTENCE COLUMN INDICATED BY THE LEFTMOST MATRIX COLUMN.

PROTOTYPES IN THE MATRIX ARE: A1 B1 C1 D1 E1 F1 G1 H1 I1 J1 K1 L1 M1 N1 O1 P1

Window / Prototype Standard Deviations = 5 columns

MATRIX 1 OF 1

MATRIX OF THE 1 TO 5 CLOSEST IDENTIFYING PROTOTYPES FOR WINDOWS OF WIDTH 32

SENTENCE COLUMNS	* GUESS 1 PROTO/DX	GUESS 2 PROTO/DX	GUESS 3 PROTO/DX	GUESS 4 PROTO/DX	GUESS 5 PROTO/DX
1- 32	* V1- .864	M1- .[illegible]	P1- .896	R1- .905	O1- .923
2- 33	* B1- .748	V1- .[illegible]	M1- .837	H1- .853	P1- .881
3- 34	* M1- .734	R1- .[illegible]	P1- .770	J1- .825	F1- .920
4- 35	* J1- .501	U1- .[illegible]	P1- .849	A1- .[illegible]	O1- .909
5- 36	* J1- .428	A1- .785	T1- .838	K1- .872	M1- .954
6- 37	* T1- .537	I1- .[illegible]	J1- .698	A1- .834	K1- .915
7- 38	* I1- .464	T1- .[illegible]	Y1- .892	A1- .949	J1-1.011
8- 39	* I1- .528	T1- .[illegible]	Y1- .870	F1- .934	A1-1.072
9- 40	* F1- .775	I1- .[illegible]	T1- .929	Y1- .935	S1- .999
10- 41	* F1- .580	P1- .[illegible]	R1- .838	E1- .834	V1- .895
11- 42	* B1- .453	P1- .[illegible]	F1- .935	E1- .649	H1- .719
12- 43	* B1- .145	R1- .[illegible]	F1- .479	E1- .531	H1- .651
13- 44	* B1- .[illegible]	R1- .[illegible]	E1- .690	O1- .790	C1- .743
14- 45	* J1- .802	R1- .[illegible]	K1- .839	B1- .855	A1- .908
15- 45	* J1- .[illegible]	A1- .879	T1- .903	K1- .915	I1- .974
16- 47	* T1- .789	I1- .[illegible]	A1- .[illegible]	J1-1.008	K1-1.038
17- 48	* T1- .826	I1- .830	A1- .948	Y1- .935	M1-1.016
18- 49	* T1- .[illegible]	I1- .959	M1- .959	Y1- .960	A1- .971
19- 50	* Y1- .866	V1- .[illegible]	N1- .934	Y1- .941	F1- .961
20- 51	* X1- .715	R1- .[illegible]	M1- .866	F1- .913	C1- .916
21- 52	* Y1- .687	T1- .[illegible]	K1- .803	R1- .[illegible]	F1- .877
22- 53	* K1- .671	T1- .[illegible]	Y1- .824	T1- .[illegible]	Y1- .927
23- 54	* K1- .729	T1- .[illegible]	T1- .[illegible]	I1- .926	Y1- .955
24- 55	* T1- .866	I1- .[illegible]	K1- .912	M1- .954	Y1-1.010
25- 56	* T1- .924	V1- .[illegible]	M1- .972	T1- .[illegible]	F1-1.056
26- 57	* C1- .828	V1- .[illegible]	F1- .[illegible]	M1-1.045	T1-1.052
27- 58	* C1- .[illegible]	V1- .835	F1- .915	U1- .918	P1- .961
28- 59	* C1- .178	U1- .[illegible]	O1- .773	D1- .824	L1- .840
29- 60	* C1- .481	O1- .[illegible]	U1- .[illegible]	D1- .791	L1- .799
30- 61	* O1- .693	U1- .[illegible]	P1- .808	C1- .[illegible]	L1- .905
31- 62	* B1- .883	O1- .903	U1-1.[illegible]	K1-1.099	W1-1.045
32- 63	* O1-1.812	O1-1.[illegible]	M1-1.[illegible]	K1-1.112	J1-1.121
33- 64	* O1-1.100	O1-1.123	G1-1.136	L1-1.144	T1-1.149
34- 65	* G1-1.028	O1-1.081	C1-1.107	H1-1.127	W1-1.172
35- 66	* G1- .932	O1-1.[illegible]	M1-1.[illegible]	D1-1.[illegible]	P1-1.127
36- 67	* G1- .926	B1- .962	M1- .959	D1-1.013	C1-1.025
37- 68	* J1- .774	R1- .[illegible]	T1- .891	A1-1.011	G1-1.019
38- 69	* J1- .[illegible]	P1- .[illegible]	T1- .910	K1- .947	P1- .969
39- 70	* J1- .477	T1- .[illegible]	A1- .[illegible]	K1- .[illegible]	T1- .[illegible]
40- 71	* T1- .[illegible]	I1- .[illegible]	J1- .731	Y1- .[illegible]	A1- .[illegible]
41- 72	* I1- .284	T1- .301	Y1- .[illegible]	A1- .[illegible]	J1-1.027
42- 73	* I1- .470	T1- .611	Y1- .931	F1- .[illegible]	S1-1.087
43- 74	* F1- .703	I1- .822	T1- .941	V1- .[illegible]	Y1- .[illegible]
44- 75	* F1- .542	C1- .[illegible]	E1- .[illegible]	P1- .[illegible]	V1- .928

```
 46- 77   * D1- .[illegible] L1- .[illegible] C1- .[illegible] H1- .831 U1- .897
 47- 78   * D1- .132 O1- .[illegible] H1- .[illegible] B1- .[illegible] L1- .743
 48- 79   * D1- .[illegible] O1- .[illegible] P1- .[illegible] B1- .[illegible] L1- .900
 49- 80   * D1- .846 J1- .961 W1-1.007 R1-1.031 F1-1.034
 50- 81   * J1- .858 A1-1.[illegible] W1-1.[illegible] R1-1.121 D1-1.130
 51- 82   * J1- .[illegible] A1- .[illegible] T1- .[illegible] L1-1.[illegible] W1-1.062
 52- 83   * T1- .872 J1- .[illegible] A1- .918 I1- .959 W1-1.100
 53- 84   * T1- .[illegible] A1- .[illegible] Y1- .912 I1- .951 J1- .964
 54- 85   * X1- .744 A1- .871 T1- .879 W1- .913 Y1- .936
 55- 86   * X1- .686 A1- .822 T1- .825 S1- .830 N1- .880
 56- 87   * X1- .769 A1- .772 T1- .739 J1- .734 K1- .810
 57- 88   * K1- .733 T1- .765 A1- .766 J1- .776 T1- .847
 58- 89   * T1- .640 I1- .725 K1- .790 A1- .832 Y1- .845
 59- 90   * I1- .603 T1- .620 Y1- .859 K1- .942 A1- .949
 60- 91   * I1- .581 T1- .77[illegible] F1- .887 V1- .905 Y1- .910
 61- 92   * F1- .718 V1- .[illegible] C1- .871 I1- .915 E1- .977
 62- 93   * F1- .563 E1- .750 C1- .750 P1- .782 V1- .813
 63- 94   * E1- .434 F1- .[illegible] B1- .611 B1- .615 U1- .695
 64- 95   * E1- .232 R1- .[illegible] L1- .[illegible] R1- .[illegible] F1- .[illegible]
 65- 96   * E1- .537 K1- .[illegible] C1- .795 H1- .800 L1- .810
 66- 97   * K1- .797 O1- .[illegible] F1- .[illegible] O1- .832 H1- .943
 67- 98   * O1- .945 O1-1.003 K1-1.020 T1-1.023 H1-1.035
 68- 99   * O1- .988 O1-1.[illegible] H1-1.[illegible] O1-1.[illegible] M1-1.089
 69-100   * O1- .974 B1- .[illegible] O1- .991 H1-1.023 C1-1.039
 70-101   * B1- .878 R1- .932 C1- .961 W1- .938 O1- .991
 71-102   * J1- .738 R1- .[illegible] F1- .881 A1- .[illegible] W1- .961
 72-103   * J1- .567 A1- .894 K1- .898 T1- .911 O1- .959
 73-104   * T1- .630 J1- .643 I1- .776 K1- .878 A1- .892
 74-105   * T1- .419 I1- .[illegible] Y1- .850 J1- .[illegible] A1- .946
 75-106   * I1- .458 T1- .[illegible] Y1- .800 F1- .963 S1-1.030
 76-107   * I1- .684 F1- .721 T1- .757 Y1- .833 S1- .941
 77-108   * F1- .427 S1- .847 F1- .[illegible] E1- .913 Y1- .927
 78-109   * F1- .253 P1- .[illegible] C1- .737 E1- .741 S1- .[illegible]
 79-110   * F1- .476 P1- .[illegible] C1- .[illegible] E1- .[illegible] O1- .685
 80-111   * B1- .628 P1- .[illegible] B1- .[illegible] O1- .[illegible] H1- .682
 81-112   * R1- .711 O1- .749 B1- .751 H1- .769 F1- .827
 82-113   * R1- .[illegible] C1- .[illegible] A1- .921 H1- .[illegible] O1- .927
 83-114   * A1- .[illegible] R1- .[illegible] J1- .[illegible] K1-1.012 P1-1.042
 84-115   * T1- .917 J1- .943 A1- .947 T1- .966 K1- .971
 85-116   * T1- .727 T1- .[illegible] I1- .922 Y1- .[illegible] K1- .947
 86-117   * T1- .634 I1- .[illegible] Y1- .[illegible] T1- .[illegible] K1- .[illegible]
 87-118   * T1- .718 I1- .[illegible] Y1- .[illegible] K1-1.[illegible] F1-1.[illegible]
 88-119   * T1- .909 I1- .913 F1-1.014 V1-1.[illegible] Y1-1.066
 89-120   * O1- .[illegible] F1- .[illegible] V1-1.[illegible] I1-1.[illegible] T1-1.[illegible]
 90-121   * O1- .762 G1- .[illegible] C1- .[illegible] F1- .[illegible] U1- .954
 91-122   * G1- .[illegible] O1- .[illegible] C1- .[illegible] J1- .[illegible] P1- .904
 92-123   * G1- .[illegible] O1- .[illegible] C1- .[illegible] H1- .[illegible] U1- .[illegible]
 93-124   * G1- .437 O1- .[illegible] C1- .787 R1- .[illegible] H1- .851
 94-125   * G1- .[illegible] R1- .[illegible] C1- .[illegible] B1- .[illegible] O1- .967
 95-126   * J1- .[illegible] K1- .[illegible] F1- .969 A1-1.[illegible] Y1-1.051
 96-127   * J1- .752 K1- .[illegible] T1- .931 Y1-1.013 A1-1.022
 97-128   * T1- .736 J1- .[illegible] I1- .[illegible] K1- .[illegible] Y1- .964
 98-129   * T1- .644 I1- .[illegible] Y1- .973 J1-1.027 W1-1.064
 99-130   * I1- .567 T1- .740 V1-1.018 Y1-1.031 H1-1.[illegible]
100-131   * I1- .[illegible] T1- .[illegible] V1- .990 F1-1.[illegible] W1-1.089
101-132   * F1- .[illegible] V1- .997 C1-1.[illegible] I1-1.103 T1-1.108
102-133   * F1- .798 P1- .944 C1- .950 L1- .951 F1- .981
```

RESULTS MATRIX SHOWING THE PROTOTYPE-TO-WINDOW
DISTANCES FOR PROTOTYPES WHOSE WIDTH VARIATION ENABLES
THE PROTOTYPE TO BE COMPARED TO WINDOWS OF WIDTH 32 COLUMNS
MATRIX IS SORTED BY DISTANCE WITHIN THE SENTENCE
COLUMNS. THE 1 TO 13 CLOSEST IDENTIFYING PROTOTYPES ARE
SHOWN FOR EACH WINDOW OF WIDTH 32 BEGINNING AND ENDING AT
THE SENTENCE COLUMN INDICATED BY THE LEFTMOST MATRIX COLUMN.

PROTOTYPES IN THE MATRIX ARE: A1 B1 C1 D1 E1 F1 G1 H1 I1 J1 K1 L1 M1 N1 O1 P1

Window / Prototype Standard Deviations = 10 columns MATRIX 1 OF 1

MATRIX OF THE 1 TO 5 CLOSEST IDENTIFYING PROTOTYPES FOR WINDOWS OF WIDTH 32

SENTENCE COLUMNS	GUESS 1 PROTO/DX	GUESS 2 PROTO/DX	GUESS 3 PROTO/DX	GUESS 4 PROTO/DX	GUESS 5 PROTO/DX
1- 32	* M1- .820	V1- .861	H1- .885	D1- .914	F1- .953
2- 33	* B1- .822	V1- .896	M1- .899	Q1- .918	F1- .927
3- 34	* B1- .836	R1- .840	W1- .859	J1- .862	F1- .971
4- 35	* J1- .718	W1- .902	F1- .939	A1- .953	B1-1.000
5- 36	* J1- .677	A1- .957	K1-1.012	T1-1.029	W1-1.039
6- 37	* T1- .831	I1- .870	J1- .879	A1-1.031	K1-1.087
7- 38	* I1- .716	T1- .756	Y1-1.116	A1-1.143	J1-1.150
8- 39	* I1- .791	T1- .980	Y1-1.076	M1-1.146	F1-1.185
9- 40	* F1- .987	I1-1.031	M1-1.069	Y1-1.116	T1-1.165
10- 41	* F1- .794	B1- .971	S1- .984	S1-1.031	V1-1.046
11- 42	* B1- .552	D1- .729	F1- .731	E1- .820	F1- .840
12- 43	* B1- .498	R1- .529	F1- .654	E1- .686	D1- .771
13- 44	* B1- .579	R1- .694	D1- .697	E1- .742	F1- .830
14- 45	* D1- .804	K1- .881	F1- .944	B1- .959	G1- .982
15- 46	* D1-1.010	K1-1.049	T1-1.089	V1-1.097	A1-1.105
16- 47	* T1-1.010	I1-1.043	M1-1.073	A1-1.123	W1-1.166
17- 48	* T1-1.050	I1-1.052	M1-1.068	A1-1.130	Y1-1.175
18- 49	* A1-1.115	M1-1.116	Y1-1.126	N1-1.151	T1-1.155
19- 50	* X1- .971	N1-1.037	F1-1.061	S1-1.066	V1-1.067
20- 51	* X1- .832	R1- .938	S1- .977	V1- .977	S1- .994
21- 52	* Y1- .795	K1- .886	T1- .902	R1- .907	F1- .912
22- 53	* K1- .724	T1- .830	Y1- .911	E1- .945	F1-1.020
23- 54	* K1- .755	T1- .983	T1- .991	E1-1.055	Y1-1.073
24- 55	* K1- .851	T1- .998	J1-1.028	M1-1.071	V1-1.119
25- 56	* I1-1.070	V1-1.073	M1-1.078	T1-1.114	C1-1.159
26- 57	* C1- .952	V1-1.031	M1-1.158	F1-1.163	U1-1.186
27- 58	* C1- .807	V1-1.053	U1-1.093	F1-1.126	T1-1.138
28- 59	* C1- .729	U1-1.003	F1-1.059	L1-1.071	S1-1.083
29- 60	* C1- .859	U1- .976	C1-1.010	L1-1.028	S1-1.056
30- 61	* O1-1.017	M1-1.044	I1-1.083	C1-1.041	F1-1.130
31- 62	* O1-1.068	O1-1.127	U1-1.153	X1-1.173	U1-1.198
32- 63	* O1-1.128	O1-1.148	M1-1.152	H1-1.190	G1-1.232
33- 64	* M1-1.046	H1-1.199	O1-1.165	O1-1.199	G1-1.195
34- 65	* M1-1.056	H1-1.118	G1-1.129	J1-1.172	O1-1.178
35- 66	* G1-1.066	H1-1.099	M1-1.183	O1-1.196	O1-1.180
36- 67	* G1-1.064	J1-1.071	I1-1.081	H1-1.115	Y1-1.127
37- 68	* J1- .896	B1- .932	F1-1.008	X1-1.056	S1-1.065
38- 69	* J1- .669	B1-1.014	F1-1.018	Z1-1.035	G1-1.057
39- 70	* J1- .640	T1- .851	I1- .996	A1- .999	W1-1.004
40- 71	* T1- .606	I1- .719	J1- .843	Y1- .953	M1-1.041
41- 72	* T1- .639	I1- .851	Y1- .987	J1-1.018	A1-1.113
42- 73	* I1- .668	T1- .736	Y1-1.014	F1-1.037	W1-1.176
43- 74	* F1- .896	I1- .948	T1-1.017	Y1-1.115	W1-1.165
44- 75	* F1- .774	G1- .897	B1-1.041	V1-1.054	U1-1.074

```
 46- 77   * D1- .875 D1- .779 L1- .781 M1- .796 W1- .807
 47- 78   * D1- .824 D1- .765 M1- .793 L1- .845 P1- .853
 48- 79   * D1- .844 C1- .891 M1- .902 B1-1.007 F1-1.024
 49- 80   * D1- .887 M1-1.040 W1-1.039 D1-1.054 W1-1.124
 50- 81   * W1-1.057 D1-1.126 M1-1.124 M1-1.172 J1-1.178
 51- 82   * W1-1.125 W1-1.138 T1-1.142 J1-1.132 A1-1.169
 52- 83   * T1-1.091 A1-1.127 B1-1.155 M1-1.158 T1-1.168
 53- 84   * X1-1.022 A1-1.068 T1-1.093 W1-1.111 W1-1.127
 54- 85   * X1- .859 A1- .938 S1- .993 W1-1.035 J1-1.037
 55- 86   * X1- .771 S1- .869 A1- .909 T1- .915 J1- .918
 56- 87   * X1- .630 J1- .832 S1- .851 T1- .855 B1- .859
 57- 88   * K1- .824 J1- .850 A1- .864 T1- .875 T1- .903
 58- 89   * T1- .809 K1- .866 J1- .906 Y1- .940 A1- .959
 59- 90   * J1- .804 T1- .824 X1-1.000 A2-1.011 K1-1.011
 60- 91   * T1- .883 V1- .956 T1- .965 F1-1.042 W1-1.043
 61- 92   * F1- .927 V1- .932 C1- .934 W1-1.037 E1-1.060
 62- 93   * F1- .825 C1- .840 F1- .935 P1- .939 W1- .942
 63- 94   * F1- .721 F1- .830 F1- .848 C1- .847 W1- .852
 64- 95   * S1- .678 L1- .795 W1- .839 P1- .834 C1- .893
 65- 96   * S1- .837 K1- .865 L1- .892 A1- .921 W1- .926
 66- 97   * D1- .918 A1- .947 C1- .966 K1- .989 W1-1.049
 67- 98   * D1- .962 W1- .984 P1- .994 D1-1.015 G1-1.141
 68- 99   * W1-1.040 D1-1.065 W1-1.050 D1-1.095 W1-1.153
 69-100   * W1-1.111 W1-1.128 F1-1.129 D1-1.133 G1-1.172
 70-101   * B1-1.091 V1-1.105 B1-1.121 D1-1.135 S1-1.171
 71-102   * J1- .952 R1-1.065 P1-1.066 A1-1.118 W1-1.126
 72-103   * J1- .791 A1-1.057 K1-1.072 T1-1.085 P1-1.104
 73-104   * J1- .792 T1- .852 J1- .962 K1-1.018 A1-1.049
 74-105   * T1- .843 I1- .741 J1- .952 Y1- .957 K1-1.067
 75-106   * T1- .651 I1- .689 Y1- .885 F1-1.055 X1-1.104
 76-107   * I1- .838 T1- .884 F1- .874 Y1- .924 X1-1.060
 77-108   * F1- .675 S1- .991 X1-1.014 X1-1.037 F1-1.038
 78-109   * F1- .582 P1- .837 P1- .873 E1- .906 S1- .956
 79-110   * F1- .689 P1- .696 P1- .710 R1- .775 W1- .781
 80-111   * W1- .572 B1- .672 F1- .681 P1- .715 P1- .720
 81-112   * W1- .715 R1- .739 D1- .760 B1- .795 P1- .873
 82-113   * W1- .891 R1- .899 D1- .940 B1- .939 K1-1.028
 83-114   * R1-1.002 W1-1.103 A1-1.113 K1-1.118 D1-1.137
 84-115   * F1-1.110 T1-1.151 X1-1.153 A1-1.136 K1-1.191
 85-116   * T1-1.047 T1-1.146 I1-1.156 Y1-1.136 X1-1.171
 86-117   * T1-1.026 I1-1.118 Y1-1.119 T1-1.186 X1-1.212
 87-118   * F1-1.051 I1-1.111 X1-1.127 F1-1.219 K1-1.245
 88-119   * T1-1.119 T1-1.140 F1-1.170 Y1-1.179 V1-1.211
 89-120   * X1-1.123 E1-1.137 V1-1.140 C1-1.158 G1-1.198
 90-121   * G1- .889 D1- .979 W1-1.033 D1-1.048 W1-1.058
 91-122   * G1- .845 D1- .820 C1- .868 W1- .886 W1- .903
 92-123   * G1- .319 D1- .698 W1- .803 D1- .815 W1- .866
 93-124   * G1- .514 D1- .708 C1- .881 R1- .844 W1- .883
 94-125   * G1- .655 R1- .690 P1- .814 B1- .975 C1- .997
 95-126   * J1- .956 K1- .999 B1-1.031 D1-1.101 A1-1.108
 96-127   * J1- .919 K1-1.010 T1-1.032 A1-1.101 Y1-1.114
 97-128   * T1- .880 I1- .971 J1- .990 Y1-1.078 K1-1.100
 98-129   * T1- .819 I1- .822 Y1-1.086 D1-1.132 B1-1.165
 99-130   * I1- .816 T1- .901 Y1-1.131 V1-1.141 F1-1.208
100-131   * I1- .964 F1-1.009 T1-1.075 V1-1.120 Y1-1.194
101-132   * F1- .904 V1-1.107 C1-1.110 E1-1.146 T1-1.151
102-133   * F1- .842 L1- .980 F1- .975 E1- .981 C1-1.002
```

Appendix G

Data Output For

Complex and Amplitude

Spectra

Sentences 1 - 4

RESULTS MATRIX SHOWING THE PROTOTYPE-TO-WINDOW
DISTANCES FOR PROTOTYPES WHOSE WIDTH VARIATION ENABLES
THE PROTOTYPE TO BE COMPARED TO WINDOWS OF WIDTH 32 COLUMNS
MATRIX IS SORTED BY DISTANCE WITHIN THE SENTENCE
COLUMNS. THE 1 TO 13 CLOSEST IDENTIFYING PROTOTYPES ARE
SHOWN FOR EACH WINDOW OF WIDTH 32 BEGINNING AND ENDING AT
THE SENTENCE COLUMN INDICATED BY THE LEFTMOST MATRIX COLUMN.

PROTOTYPES IN THE MATRIX ARE: A1 B1 C1 D1 E1 F1 G1 H1 I1 J1 K1 L1 M1 N1 O1 P1 Q

Complex Spectrum : Sentence #1

MATRIX 1 OF 1

MATRIX OF THE 1 TO 5 CLOSEST IDENTIFYING PROTOTYPES FOR WINDOWS OF WIDTH 32 S

SENTENCE COLUMNS	*	GUESS 1 PROTO/DX	GUESS 2 PROTO/DX	GUESS 3 PROTO/DX	GUESS 4 PROTO/DX	GUESS 5 PROTO/DX
1- 32	*	V1- .624	M1- .732	C1- .815	B1- .823	H1- .847
2- 33	*	B1- .695	R1- .744	C1- .759	H1- .792	V1- .812
3- 34	*	R1- .730	B1- .731	V1- .808	P1- .879	M1- .880
4- 35	*	J1- .731	W1- .776	A1- .846	R1- .848	P1- .913
5- 36	*	J1- .766	W1- .810	A1- .810	T1- .852	I1- .902
6- 37	*	J1- .769	T1- .803	I1- .856	A1- .869	W1- .893
7- 38	*	I1- .398	T1- .708	J1- .859	Y1- .920	A1- .923
8- 39	*	I1- .470	T1- .856	Y1- .855	J1- .912	A1- .921
9- 40	*	I1- .768	Y1- .849	F1- .891	V1- .895	T1- .906
10- 41	*	F1- .692	E1- .716	C1- .778	L1- .785	V1- .828
11- 42	*	B1- .452	E1- .536	F1- .570	P1- .614	C1- .647
12- 43	*	R1- .293	B1- .513	F1- .559	E1- .563	M1- .634
13- 44	*	B1- .505	R1- .632	F1- .738	D1- .720	S1- .734
14- 45	*	W1- .840	S1- .842	J1- .853	B1- .872	C1- .872
15- 46	*	J1- .774	A1- .877	W1- .878	T1- .887	S1- .958
16- 47	*	T1- .803	J1- .867	I1- .821	A1- .884	W1- .926
17- 48	*	I1- .804	T1- .843	A1- .882	J1- .905	W1- .949
18- 49	*	A1- .863	I1- .908	V1- .916	W1- .922	Y1- .943
19- 50	*	X1- .822	A1- .838	W1- .851	Y1- .887	V1- .902
20- 51	*	X1- .678	W1- .776	A1- .815	R1- .843	M1- .854
21- 52	*	X1- .843	W1- .736	K1- .739	Z1- .810	A1- .820
22- 53	*	K1- .718	X1- .736	W1- .781	Z1- .827	A1- .857
23- 54	*	K1- .799	F1- .822	D1- .834	V1- .883	Y1- .911
24- 55	*	I1- .837	F1- .862	K1- .923	V1- .933	Y1- .956
25- 56	*	V1- .890	T1- .892	T1- .989	L1- .989	L1-1.009
26- 57	*	C1- .934	V1- .872	I1- .894	E1- .931	F1- .939
27- 58	*	C1- .812	L1- .816	U1- .829	E1- .842	F1- .855
28- 59	*	C1- .866	U1- .713	G1- .769	E1- .791	F1- .809
29- 60	*	C1- .551	G1- .707	U1- .723	D1- .745	O1- .804
30- 61	*	G1- .779	C1- .786	L1- .790	U1- .851	D1- .859
31- 62	*	D1- .868	G1- .917	O1- .918	R1-1.015	K1-1.016
32- 63	*	O1- .945	D1- .996	G1-1.022	J1-1.059	W1-1.090
33- 64	*	D1- .962	G1-1.045	O1-1.046	J1-1.053	T1-1.070
34- 65	*	O1- .938	G1-1.007	O1-1.041	H1-1.057	O1-1.085
35- 66	*	O1- .931	G1- .969	H1- .993	D1-1.008	O1-1.011
36- 67	*	R1- .903	R1- .935	O1- .933	H1- .934	O1- .966
37- 68	*	J1- .878	B1- .878	I1- .879	A1- .887	S1- .940
38- 69	*	J1- .859	A1- .895	S1- .915	R1- .929	T1- .929
39- 70	*	J1- .869	T1- .898	I1- .861	W1- .889	T1- .909
40- 71	*	T1- .832	J1- .877	I1- .891	Y1- .893	A1- .919
41- 72	*	I1- .267	T1- .825	J1- .891	Y1- .926	T1- .949
42- 73	*	I1- .829	T1- .815	Y1- .851	J1- .845	T1- .987
43- 74	*	I1- .758	F1- .852	Y1- .881	L1- .892	T1- .906

```
 46- 77   * D1- .527 U1- .[illegible] E1- .595 B1- .[illegible] C1- .613
 47- 78   * D1- .[illegible] O1- .[illegible] P1- .639 B1- .[illegible] U1- .[illegible]
 48- 79   * D1- .[illegible] O1- .736 P1- .807 R1- .[illegible] T1- .832
 49- 80   * D1- .[illegible] O1- .[illegible] P1- .[illegible] A1- .[illegible] S1- .999
 50- 81   * J1- .896 W1-1.[illegible] A1-1.017 D1-1.[illegible] E1-1.101
 51- 82   * J1- .797 A1- .[illegible] W1- .[illegible] T1- .[illegible] E1-1.080
 52- 83   * J1- .759 T1- .[illegible] U1- .832 A1- .[illegible] T1- .[illegible]
 53- 84   * J1- .787 W1- .[illegible] T1- .[illegible] A1- .[illegible] Y1- .894
 54- 85   * X1- .720 W1- .[illegible] E1- .806 J1- .[illegible] T1- .[illegible]
 55- 86   * X1- .631 A1- .746 T1- .786 W1- .[illegible] Y1- .807
 56- 87   * X1- .689 A1- .699 S1- .762 J1- .[illegible] Y1- .779
 57- 88   * A1- .699 J1- .722 T1- .751 Y1- .[illegible] S1- .750
 58- 89   * T1- .[illegible] I1- .[illegible] J1- .737 A1- .[illegible] Y1- .791
 59- 90   * I1- .[illegible] T1- .[illegible] Y1- .805 J1- .[illegible] E1- .849
 60- 91   * I1- .[illegible] E1- .774 Y1- .836 W1- .[illegible] E1- .[illegible]
 61- 92   * E1- .[illegible] E1- .[illegible] W1- .817 L1- .[illegible] C1- .[illegible]
 62- 93   * E1- .[illegible] E1- .[illegible] I1- .[illegible] O1- .[illegible] B1- .764
 63- 94   * E1- .[illegible] E1- .[illegible] B1- .801 L1- .[illegible] E1- .[illegible]
 64- 95   * E1- .671 E1- .[illegible] E1- .[illegible] B1- .[illegible] W1- .528
 65- 96   * S1- .[illegible] K1- .[illegible] E1- .736 E1- .[illegible] B1- .794
 66- 97   * K1- .[illegible] O1- .[illegible] E1- .[illegible] D1- .[illegible] T1- .[illegible]
 67- 98   * O1- .[illegible] O1- .922 W1- .[illegible] T1- .[illegible] S1- .983
 68- 99   * O1- .[illegible] O1- .927 O1- .[illegible] W1- .[illegible] O1- .978
 69-100   * O1- .[illegible] O1- .[illegible] T1- .[illegible] D1- .[illegible] E1- .941
 70-101   * B1- .[illegible] R1- .[illegible] S1- .[illegible] D1- .[illegible] O1- .[illegible]
 71-102   * J1- .795 B1- .841 E1- .887 S1- .[illegible] A1- .899
 72-103   * J1- .[illegible] T1- .[illegible] W1- .[illegible] A1- .[illegible] S1- .[illegible]
 73-104   * J1- .[illegible] T1- .[illegible] I1- .[illegible] W1- .[illegible] A1- .[illegible]
 74-105   * T1- .[illegible] I1- .[illegible] J1- .[illegible] Y1- .[illegible] T1- .920
 75-106   * I1- .[illegible] T1- .[illegible] Y1- .[illegible] J1- .[illegible] T1- .919
 76-107   * I1- .[illegible] T1- .[illegible] Y1- .740 E1- .[illegible] S1- .[illegible]
 77-108   * E1- .[illegible] E1- .[illegible] Y1- .731 S1- .[illegible] B1- .879
 78-109   * E1- .[illegible] E1- .[illegible] E1- .[illegible] B1- .[illegible] S1- .723
 79-110   * E1- .[illegible] B1- .[illegible] E1- .[illegible] B1- .[illegible] C1- .[illegible]
 80-111   * B1- .[illegible] B1- .[illegible] E1- .[illegible] D1- .[illegible] H1- .[illegible]
 81-112   * R1- .[illegible] B1- .[illegible] C1- .[illegible] B1- .[illegible] H1- .[illegible]
 82-113   * R1- .[illegible] A1- .[illegible] B1- .[illegible] K1- .[illegible] H1- .[illegible]
 83-114   * A1- .[illegible] W1- .912 K1- .[illegible] R1- .[illegible] J1- .[illegible]
 84-115   * T1- .892 J1- .[illegible] A1- .913 W1- .[illegible] K1- .[illegible]
 85-116   * T1- .[illegible] I1- .[illegible] J1- .[illegible] Y1- .[illegible] T1- .[illegible]
 86-117   * T1- .[illegible] I1- .[illegible] Y1- .[illegible] J1- .[illegible] T1- .[illegible]
 87-118   * T1- .[illegible] I1- .[illegible] Y1- .[illegible] V1-1.[illegible] W1-1.[illegible]
 88-119   * I1- .[illegible] T1- .[illegible] Y1- .[illegible] V1- .[illegible] L1-1.[illegible]
 89-120   * L1- .[illegible] C1- .[illegible] E1- .[illegible] B1- .[illegible] V1- .[illegible]
 90-121   * C1- .[illegible] G1- .[illegible] L1- .[illegible] D1- .[illegible] E1- .[illegible]
 91-122   * G1- .[illegible] C1- .[illegible] W1- .[illegible] D1- .[illegible] L1- .[illegible]
 92-123   * G1- .[illegible] O1- .[illegible] B1- .[illegible] C1- .[illegible] W1- .[illegible]
 93-124   * G1- .[illegible] O1- .[illegible] B1- .[illegible] H1- .[illegible] E1- .[illegible]
 94-125   * S1- .[illegible] B1- .[illegible] E1- .[illegible] G1- .[illegible] B1- .[illegible]
 95-126   * J1- .[illegible] S1- .[illegible] E1- .[illegible] X1- .[illegible] B1-1.[illegible]
 96-127   * J1- .[illegible] T1- .[illegible] S1-1.[illegible] A1-1.[illegible] Y1-1.[illegible]
 97-128   * J1- .[illegible] T1- .[illegible] J1- .[illegible] Y1- .[illegible] B1-1.[illegible]
 98-129   * I1- .[illegible] T1- .[illegible] J1- .[illegible] Y1- .[illegible] W1-1.[illegible]
 99-130   * I1- .[illegible] T1- .[illegible] J1- .[illegible] Y1-1.[illegible] W1-1.[illegible]
100-131   * I1- .[illegible] T1- .[illegible] W1-1.[illegible] V1-1.[illegible] T1-1.[illegible]
101-132   * L1- .[illegible] F1- .[illegible] E1- .[illegible] Z1-1.[illegible] V1-1.[illegible]
102-133   * L1- .[illegible] F1- .[illegible] F1- .[illegible] P1- .[illegible] U1- .[illegible]
```

RESULTS MATRIX SHOWING THE PROTOTYPE-TO-WINDOW
DISTANCES FOR PROTOTYPES WHOSE WIDTH VARIATION ENABLES
THE PROTOTYPE TO BE COMPARED TO WINDOWS OF WIDTH 32 COLUMNS
MATRIX IS SORTED BY DISTANCE WITHIN THE SENTENCE
COLUMNS. THE 1 TO 13 CLOSEST IDENTIFYING PROTOTYPES ARE
SHOWN FOR EACH WINDOW OF WIDTH 32 BEGINNING AND ENDING AT
THE SENTENCE COLUMN INDICATED BY THE LEFTMOST MATRIX COLUMN.

PROTOTYPES IN THE MATRIX ARE: A1 B1 C1 D1 E1 F1 G1 H1 I1 J1 K1 L1 M1 N1 O1 P1

Complex Spectrum - Sentence # 2

MATRIX 1 OF 1

MATRIX OF THE 1 TO 5 CLOSEST IDENTIFYING PROTOTYPES FOR WINDOWS OF WIDTH 32

SENTENCE COLUMNS	GUESS 1 PROTO/DX	GUESS 2 PROTO/DX	GUESS 3 PROTO/DX	GUESS 4 PROTO/DX	GUESS 5 PROTO/DX
1- 32	J1- .667	T1- .690	J1- .710	W1- .938	A1- .971
2- 33	I1- .649	T1- .682	J1- .717	Y1- .971	W1- .982
3- 34	I1- .616	T1- .662	Y1- .925	J1- .938	W1-1.009
4- 35	I1- .818	V1- .900	T1- .932	Y1- .932	S1- .941
5- 36	E1- .730	B1- .740	F1- .773	S1- .818	D1- .825
6- 37	B1- .620	E1- .686	A1- .690	F1- .677	D1- .685
7- 38	R1- .572	B1- .619	E1- .653	K1- .668	D1- .698
8- 39	J1- .743	K1- .773	F1- .787	W1- .813	A1- .813
9- 40	T1- .669	J1- .676	I1- .796	W1- .821	A1- .850
10- 41	I1- .653	T1- .659	J1- .794	W1- .891	A1- .914
11- 42	I1- .576	T1- .718	Y1- .937	W1- .979	A1- .983
12- 43	I1- .815	V1- .910	L1- .921	Y1- .927	T1- .939
13- 44	L1- .774	V1- .794	B1- .865	E1- .887	W1- .929
14- 45	V1- .741	L1- .745	L1- .757	E1- .820	D1- .845
15- 46	J1- .661	O1- .769	V1- .771	W1- .792	D1- .792
16- 47	J1- .711	O1- .719	C1- .727	W1- .737	C1- .785
17- 48	D1- .778	O1- .798	B1- .800	J1- .813	C1- .823
18- 49	R1- .886	B1- .895	W1- .913	D1- .913	G1- .947
19- 50	J1- .808	W1- .907	B1- .917	A1- .955	D1- .974
20- 51	J1- .568	A1- .884	W1- .917	T1- .971	C1-1.028
21- 52	J1- .673	T1- .733	I1- .819	A1- .890	W1- .946
22- 53	I1- .843	T1- .871	J1- .880	A1- .951	W1- .993
23- 54	I1- .611	T1- .662	J1- .932	Y1- .932	B1-1.024
24- 55	I1- .786	T1- .921	Y1- .979	W1-1.027	Y1-1.030
25- 56	E1- .882	F1- .884	V1- .907	L1- .925	C1- .955
26- 57	B1- .667	E1- .702	A1- .761	F1- .773	W1- .781
27- 58	B1- .627	R1- .690	B1- .871	E1- .880	A1- .717
28- 59	R1- .878	B1- .736	F1- .776	D1- .802	W1- .869
29- 60	J1- .784	W1- .890	B1- .923	K1- .948	A1- .964
30- 61	J1- .845	T1- .706	I1- .858	W1- .910	A1- .964
31- 62	I1- .587	T1- .653	J1- .720	W1- .950	A1- .989
32- 63	I1- .633	T1- .691	J1- .903	Y1- .962	A1- .992
33- 64	I1- .785	Y1- .919	T1- .922	S1- .962	A1- .962
34- 65	S1- .776	X1- .777	B1- .805	F1- .805	B1- .836
35- 66	X1- .622	E1- .845	S1- .865	F1- .887	B1- .889
36- 67	K1- .560	X1- .648	B1- .678	R1- .715	S1- .714
37- 68	K1- .602	X1- .642	B1- .850	E1- .865	S1- .881
38- 69	T1- .806	K1- .821	Y1- .927	I1- .950	J1- .973
39- 70	T1- .818	I1- .867	M1- .927	V1- .929	Y1- .964
40- 71	V1- .876	M1- .911	I1- .953	T1- .987	C1-1.017
41- 72	O1- .872	C1- .872	V1- .930	D1- .937	G1- .938
42- 73	O1- .788	R1- .789	G1- .798	C1- .805	C1- .825
43- 74	B1- .670	R1- .722	G1- .810	H1- .925	C1- .828
44- 75	R1- .764	B1- .795	J1- .830	W1- .838	A1- .916

[illegible]		J1- .617	A1- [illegible]	Y1- .900	K1- [illegible]	[illegible] .930
46- 77	*	J1- .866	T1- .836	I1- .870	A1- .931	W1- .935
47- 78	*	T1- .910	T1- .903	J1- .930	Y1- .931	A1- .959
48- 79	*	I1- .389	T1- .485	J1- .649	Y1- .854	71-1.002
49- 80	*	I1- .677	T1- .771	Y1- .852	V1- .937	71-1.006
50- 81	*	F1- .501	E1- .517	V1- .533	O1- .579	L1- .586
51- 82	*	E1- .618	F1- .628	R1- .687	P1- .712	C1- .728
52- 83	*	P1- .497	R1- .567	R1- .566	E1- .515	F1- .641
53- 84	*	P1- .579	O1- .612	F1- .653	31- .718	E1- .608
54- 85	*	O1- .795	P1- .834	F1- .889	J1- .940	P1- .950
55- 86	*	J1- .863	W1- .951	A1- .988	T1- .994	O1-1.026
56- 87	*	J1- .868	A1- .886	I1- .934	W1- .939	T1- .951
57- 88	*	A1- .791	I1- .910	J1- .921	W1- .921	T1- .961
58- 89	*	A1- .754	W1- .926	M1- .943	I1- .950	X1- .952
59- 90	*	X1- .742	N1- .755	A1- .813	S1- .953	M1- .975
60- 91	*	X1- .544	N1- .660	Y1- .910	A1- .941	C1- .952
61- 92	*	X1- .739	N1- .743	Y1- .789	41- .914	K1- .936
62- 93	*	M1- .670	V1- .794	Y1- .769	N1- .907	X1- .930
63- 94	*	M1- .558	V1- .647	Y1- .872	K1- .938	H1-1.016
64- 95	*	M1- .586	V1- .706	H1- .915	R1- .935	P1- .995
65- 96	*	M1- .818	H1- .857	W1- .870	31- .874	F1- .879
66- 97	*	W1- .775	31- .826	F1- .639	H1- .913	71- .952
67- 98	*	W1- .756	A1- .847	F1- .690	31- .917	J1- .921
68- 99	*	J1- .690	A1- .790	W1- .837	T1- .949	K1- .962
69-100	*	J1- .566	T1- .693	I1- .816	A1- .825	W1- .890
70-101	*	T1- .517	I1- .532	J1- .664	A1- .915	Y1- .941
71-102	*	I1- .472	T1- .604	Y1- .656	J1- .833	A1-1.007
72-103	*	I1- .716	Y1- .839	T1- .851	V1- .898	F1- .946
73-104	*	F1- .751	E1- .776	C1- .810	V1- .828	P1- .863
74-105	*	P1- .520	E1- .624	F1- .627	31- .631	F1- .654
75-106	*	P1- .521	R1- .578	F1- .619	E1- .675	F1- .701
76-107	*	P1- .596	R1- .766	C1- .792	31- .840	K1- .857
77-108	*	J1- .913	W1- .926	F1- .950	O1- .975	A1- .961
78-109	*	T1- .914	J1- .930	I1- .936	W1- .951	A1- .976
79-110	*	I1- .902	T1- .935	A1- .962	W1- .937	J1-1.018
80-111	*	A1- .943	I1- .981	W1- .995	T1-1.019	M1-1.093
81-112	*	N1- .889	A1- .935	Y1- .975	W1- .998	S1-1.066
82-113	*	N1- .641	X1- .776	S1- .944	A1- .953	91- .995
83-114	*	N1- .470	X1- .632	S1- .851	31- .953	Y1- .965
84-115	*	N1- .558	X1- .645	S1- .828	K1- .911	M1- .919
85-116	*	N1- .805	X1- .814	M1- .842	K1- .871	S1- .884
86-117	*	M1- .827	K1- .916	V1- .922	R1- .931	Y1- .984
87-118	*	M1- .871	V1- .906	K1-1.011	W1-1.013	F1-1.025
88-119	*	V1- .924	M1- .945	W1- .962	A1-1.030	J1-1.032
89-120	*	W1- .897	A1- .946	V1- .965	J1-1.024	M1-1.024
90-121	*	W1- .825	A1- .856	F1- .977	31- .937	J1- .999
91-122	*	W1- .769	A1- .735	F1- .907	X1- .943	J1- .945
92-123	*	W1- .757	X1- .776	A1- .785	J1- .873	C1- .883
93-124	*	X1- .732	W1- .802	71- .806	K1- .812	J1- .841
94-125	*	T1- .772	K1- .804	71- .817	Y1- .823	Y1- .832
95-126	*	T1- .729	I1- .736	Y1- .826	K1- .833	71- .916
96-127	*	I1- .795	T1- .816	Y1- .901	V1- .921	K1- .981
97-128	*	V1- .891	I1- .916	L1- .950	F1- .970	E1- .974
98-129	*	C1- .796	L1- .837	E1- .873	F1- .884	V1- .931
99-130	*	C1- .551	U1- .756	L1- .774	E1- .737	G1- .802
100-131	*	C1- .436	G1- .575	U1- .637	O1- .642	C1- .768
101-132	*	O1- .414	G1- .506	C1- .602	O1- .643	U1- .576
102-133	*	O1- .502	O1- .596	G1- .663	O1- .765	H1- .820

ESULTS MATRIX SHOWING THE PROTOTYPE-TO-WINDOW
ISTANCES FOR PROTOTYPES WHOSE WIDTH VARIATION ENABLES
E PROTOTYPE TO BE COMPARED TO WINDOWS OF WIDTH 32 COLUMNS
IRIX IS SORTED BY DISTANCE WITHIN THE SENTENCE
OLUMNS. THE 1 TO 13 CLOSEST IDENTIFYING PROTOTYPES ARE
OWN FOR EACH WINDOW OF WIDTH 32 BEGINNING AND ENDING AT
HE SENTENCE COLUMN INDICATED BY THE LEFTMOST MATRIX COLUMN.

ROTOTYPES IN THE MATRIX ARE: A1 B1 C1 D1 E1 F1 G1 H1 I1 J1 K1 L1 M1 N1 O1 P1 Q1

Complex Spectrum - Sentence #3

MATRIX 1 OF 1

ATRIX OF THE 1 TO 5 CLOSEST IDENTIFYING PROTOTYPES FOR WINDOWS OF WIDTH 32 SO

SENTENCE COLUMNS	*	GUESS 1 PROTO/DX	GUESS 2 PROTO/DX	GUESS 3 PROTO/DX	GUESS 4 PROTO/DX	GUESS 5 PROTO/DX
1- 32	*	X1- .741	A1- .85[illegible]	M1- .857	S1- .881	T1- .896
2- 33	*	X1- .824	T1- .83[illegible]	Y1- .847	I1- .891	S1- .919
3- 34	*	I1- .837	T1- .84[illegible]	Y1- .852	X1- .855	S1- .873
4- 35	*	I1- .863	Y1- .906	V1- .908	T1- .925	L1- .959
5- 36	*	V1- .834	L1- .863	C1- .965	E1- .963	J1- .968
6- 37	*	C1- .789	L1- .836	O1- .820	V1- .824	U1- .813
7- 38	*	O1- .682	C1- .702	U1- .729	S1- .822	L1- .835
8- 39	*	O1- .494	U1- .731	C1- .757	G1- .761	O1- .801
9- 40	*	O1- .615	G1- .819	C1- .836	D1- .865	U1- .871
10- 41	*	O1- .859	R1- .921	O1- .943	S1- .955	O1- .967
11- 42	*	J1- .831	R1- .998	W1-1.003	P1-1.052	O1-1.087
12- 43	*	J1- .707	W1- .962	T1- .966	A1-1.043	T1-1.060
13- 44	*	J1- .706	T1- .946	W1- .929	I1- .934	T1- .953
14- 45	*	J1- .819	T1- .829	T1- .859	I1- .868	A1- .893
15- 46	*	T1- .817	A1- .832	X1- .883	T1- .918	S1- .928
16- 47	*	A1- .774	X1- .788	T1- .834	S1- .843	T1- .876
17- 48	*	A1- .730	B1- .764	T1- .767	X1- .777	S1- .801
18- 49	*	A1- .746	R1- .772	J1- .793	S1- .811	P1- .824
19- 50	*	J1- .671	T1- .762	A1- .810	W1- .854	Y1- .867
20- 51	*	T1- .653	I1- .851	J1- .875	Y1- .885	W1- .886
21- 52	*	I1- .483	T1- .913	J1- .833	Y1- .880	W1- .905
22- 53	*	I1- .608	T1- .719	Y1- .835	V1- .868	W1- .957
23- 54	*	F1- .830	V1- .872	I1- .894	E1- .902	C1- .927
24- 55	*	F1- .666	J1- .710	S1- .729	P1- .733	C1- .816
25- 56	*	B1- .659	F1- .[illegible]06	F1- .613	I1- .620	P1- .625
26- 57	*	B1- .502	R1- .[illegible]20	F1- .625	E1- .689	H1- .703
27- 58	*	B1- .689	K1- .791	F1- .799	S1- .812	W1- .826
28- 59	*	J1- .819	K1- .856	W1- .858	A1- .858	L1- .931
29- 60	*	T1- .747	J1- .812	J1- .861	A1- .910	W1- .934
30- 61	*	T1- .698	I1- .702	J1- .927	A1- .952	Y1-1.000
31- 62	*	I1- .746	T1- .920	V1- .912	Y1- .924	M1-1.019
32- 63	*	V1- .797	I1- .927	Y1- .929	M1- .932	T1-1.001
33- 64	*	V1- .787	Y1- .945	E1- .952	L1- .955	B1- .967
34- 65	*	V1- .882	W1- .895	F1- .895	B1- .904	E1- .906
35- 66	*	W1- .831	P1- .928	E1- .948	F1- .950	B1- .951
36- 67	*	W1- .822	J1- .958	T1- .980	A1- .984	F1-1.009
37- 68	*	W1- .868	T1- .931	J1- .965	V1- .962	A1- .961
38- 69	*	W1- .831	S1- .896	W1- .935	X1- .974	T1- .975
39- 70	*	S1- .703	N1- .842	Y1- .843	B1- .915	F1- .939
40- 71	*	S1- .568	X1- .772	E1- .837	V1- .874	F1- .875
41- 72	*	S1- .548	X1- .820	F1- .894	E1- .898	V1- .976
42- 73	*	S1- .704	K1- .856	F1- .934	E1- .953	Y1- .961
43- 74	*	S1- .965	K1- .905	T1- .924	J1- .940	T1- .942
44- 75	*	T1- .879	I1- .904	V1-1.012	J1-1.032	M1-1.043

[illegible]	*	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
46- 77	*	I1-1.016	C1-1.037	M1-1.052	T1-1.057	F1-1.085
47- 78	*	G1- .914	D1- .937	C1- .955	J1- .959	M1- .968
48- 79	*	H1- .796	D1- .818	F1- .826	G1- .827	P1- .856
49- 80	*	P1- .714	H1- .743	F1- .752	D1- .739	G1- .821
50- 81	*	B1- .773	R1- .781	M1- .863	D1- .878	G1- .944
51- 82	*	J1- .551	R1- .926	A1- .930	B1- .933	W1- .959
52- 83	*	J1- .537	T1- .818	A1- .923	W1- .935	Z1- .996
53- 84	*	I1- .514	J1- .566	I1- .731	Y1- .910	W1- .945
54- 85	*	T1- .329	I1- .455	J1- .656	Y1- .814	W1- .978
55- 86	*	I1- .438	T1- .507	Y1- .769	J1- .903	V1- .998
56- 87	*	I1- .701	T1- .806	Y1- .805	V1- .903	F1- .943
57- 88	*	F1- .769	E1- .843	C1- .860	V1- .852	L1- .890
58- 89	*	F1- .668	P1- .695	E1- .720	D1- .721	P1- .733
59- 90	*	P1- .562	B1- .577	D1- .600	H1- .640	P1- .656
60- 91	*	D1- .588	B1- .615	M1- .633	R1- .633	P1- .641
61- 92	*	R1- .764	B1- .794	D1- .734	H1- .817	P1- .846
62- 93	*	A1- .811	J1- .820	M1- .875	R1- .902	K1- .990
63- 94	*	J1- .678	A1- .791	I1- .868	W1- .937	Z1-1.015
64- 95	*	T1- .667	J1- .659	I1- .771	A1- .837	W1- .932
65- 96	*	T1- .520	I1- .531	J1- .763	Y1- .834	A1- .927
66- 97	*	I1- .465	T1- .604	Y1- .792	J1- .977	X1-1.006
67- 98	*	I1- .651	Y1- .766	I1- .818	V1- .875	L1- .970
68- 99	*	L1- .757	V1- .763	Y1- .831	F1- .855	F1- .867
69-100	*	L1- .564	U1- .717	F1- .727	F1- .728	C1- .750
70-101	*	U1- .450	L1- .510	C1- .615	E1- .635	F1- .665
71-102	*	U1- .342	C1- .615	L1- .657	H1- .670	G1- .678
72-103	*	U1- .560	D1- .576	G1- .680	H1- .712	L1- .706
73-104	*	D1- .789	D1- .811	G1- .827	D1- .855	U1- .874
74-105	*	J1- .967	D1- .986	C1-1.005	R1-1.021	G1-1.022
75-106	*	J1- .793	T1- .980	M1-1.075	S1-1.119	Z1-1.129
76-107	*	J1- .732	T1- .828	D1- .957	W1-1.037	Z1-1.123
77-108	*	T1- .783	J1- .835	I1- .842	Z1-1.111	U1-1.113
78-109	*	T1- .891	I1- .893	J1-1.009	Z1-1.002	Y1-1.103
79-110	*	D1-1.040	P1-1.046	D1-1.037	T1-1.057	Z1-1.070
80-111	*	P1- .922	D1- .971	B1- .991	D1- .995	M1-1.007
81-112	*	P1- .878	B1- .902	D1- .917	R1- .949	M1- .989
82-113	*	B1- .916	P1- .950	F1- .971	D1- .975	A1- .991
83-114	*	A1- .828	W1- .937	J1- .972	B1-1.014	P1-1.047
84-115	*	A1- .697	J1- .864	W1- .874	T1-1.031	U1-1.055
85-116	*	A1- .675	J1- .824	W1- .866	T1- .935	U1- .935
86-117	*	A1- .776	T1- .805	Y1- .828	I1- .834	F1- .853
87-118	*	X1- .777	I1- .736	I1- .730	Y1- .845	M1- .854
88-119	*	Y1- .669	I1- .819	V1- .830	X1- .837	I1- .849
89-120	*	V1- .527	Y1- .602	M1- .843	I1- .933	I1- .946
90-121	*	V1- .323	Y1- .687	M1- .807	F1- .915	F1- .945
91-122	*	V1- .496	F1- .819	Y1- .847	P1- .852	L1- .876
92-123	*	F1- .783	P1- .736	V1- .814	W1- .832	E1- .843
93-124	*	W1- .733	F1- .820	F1- .822	R1- .857	E1- .870
94-125	*	W1- .770	Z1- .866	F1- .912	P1- .925	K1- .950
95-126	*	W1- .917	Z1- .942	J1-1.005	F1-1.010	A1-1.016
96-127	*	A1- .978	I1-1.012	J1-1.049	T1-1.073	F1-1.074
97-128	*	A1- .992	I1- .994	F1-1.089	D1-1.123	[illegible]1-1.125
98-129	*	I1-1.057	A1-1.061	F1-1.065	D1-1.031	L1-1.096
99-130	*	F1-1.027	L1-1.036	C1-1.039	E1-1.052	P1-1.109
100-131	*	L1- .968	F1-1.007	C1-1.037	E1-1.035	U1-1.065
101-132	*	L1- .991	U1-1.026	F1-1.029	E1-1.056	P1-1.067
102-133	*	U1-1.039	L1-1.042	F1-1.091	P1-1.104	P1-1.109

RESULTS MATRIX SHOWING THE PROTOTYPE-TO-WINDOW
DISTANCES FOR PROTOTYPES WHOSE WIDTH VARIATION ENABLES
THE PROTOTYPE TO BE COMPARED TO WINDOWS OF WIDTH 32 COLUMNS
MATRIX IS SORTED BY DISTANCE WITHIN THE SENTENCE
COLUMNS. THE 1 TO 13 CLOSEST IDENTIFYING PROTOTYPES ARE
SHOWN FOR EACH WINDOW OF WIDTH 32 BEGINNING AND ENDING AT
THE SENTENCE COLUMN INDICATED BY THE LEFTMOST MATRIX COLUMN.

PROTOTYPES IN THE MATRIX ARE: A1 B1 C1 D1 E1 F1 G1 H1 I1 J1 K1 L1 M1 N1 O1 P1 Q

Complex Spectrum – Sentence "4

MATRIX 1 OF 1

MATRIX OF THE 1 TO 5 CLOSEST IDENTIFYING PROTOTYPES FOR WINDOWS OF WIDTH 32 S

SENTENCE COLUMNS	* GUESS 1 PROTO/DX	GUESS 2 PROTO/DX	GUESS 3 PROTO/DX	GUESS 4 PROTO/DX	GUESS 5 PROTO/DX
1- 32	* W1- .738	P1- .840	Z1- .872	R1- .883	O1- .913
2- 33	* W1- .749	A1- .844	Z1- .883	J1- .895	F1- .918
3- 34	* A1- .643	W1- .799	J1- .807	Z1- .955	S1- .997
4- 35	* A1- .533	J1- .778	W1- .874	S1- .958	B1- .971
5- 36	* A1- .608	J1- .826	I1- .863	T1- .887	M1- .899
6- 37	* A1- .808	T1- .862	X1- .819	T1- .840	N1- .854
7- 38	* Y1- .766	X1- .819	J1- .821	T1- .855	M1- .883
8- 39	* Y1- .601	V1- .671	M1- .879	X1- .888	B1- .915
9- 40	* V1- .472	Y1- .601	M1- .798	F1- .884	K1- .938
10- 41	* V1- .541	Y1- .767	F1- .831	M1- .845	F1- .891
11- 42	* V1- .803	F1- .822	F1- .862	E1- .858	W1- .876
12- 43	* W1- .769	F1- .878	C1- .893	Z1- .907	F1- .908
13- 44	* W1- .794	C1- .912	Z1- .947	E1- .948	K1- .962
14- 45	* C1- .903	W1- .934	G1- .956	O1- .972	F1- .994
15- 45	* C1- .888	O1- .893	G1- .921	E1-1.007	G1-1.035
16- 47	* O1- .839	C1- .870	G1- .938	O1- .973	F1- .995
17- 48	* O1- .842	C1- .912	F1- .916	S1- .941	G1- .960
18- 49	* O1- .894	O1- .911	G1- .970	R1- .979	F1- .990
19- 50	* O1- .945	J1-1.016	F1-1.011	O1-1.021	F1-1.025
20- 51	* J1- .923	T1-1.042	E1-1.052	R1-1.060	Z1-1.065
21- 52	* J1- .873	T1- .972	O1-1.003	A1-1.052	W1-1.062
22- 53	* J1- .892	T1- .937	I1- .984	A1- .952	W1-1.007
23- 54	* A1- .883	J1- .938	T1- .949	E1- .955	W1- .965
24- 55	* A1- .849	N1- .890	X1- .838	W1- .945	Z1- .962
25- 56	* X1- .581	N1- .660	Z1- .804	S1- .818	W1- .947
26- 57	* X1- .310	N1- .570	S1- .815	Y1- .825	W1- .904
27- 58	* X1- .633	N1- .712	Y1- .755	K1- .760	S1- .818
28- 59	* K1- .725	X1- .756	Y1- .736	S1- .811	W1- .917
29- 60	* K1- .820	V1- .856	Y1- .878	T1- .933	M1- .978
30- 61	* V1- .842	T1- .908	J1- .937	K1- .973	Y1- .985
31- 62	* V1- .878	I1- .913	T1- .935	O1-1.005	L1-1.059
32- 63	* C1- .888	V1- .946	I1- .955	L1-1.015	F1-1.029
33- 64	* C1- .733	U1- .944	F1- .953	O1- .958	L1- .973
34- 65	* C1- .550	U1- .848	F1- .886	G1- .901	O1- .902
35- 66	* C1- .693	U1- .818	G1- .850	F1- .850	O1- .890
36- 67	* C1- .830	O1- .848	G1- .874	J1- .882	F1- .894
37- 68	* O1- .853	P1- .915	C1- .918	G1- .937	F1- .981
38- 69	* O1- .810	O1- .921	F1- .952	R1-1.023	G1-1.027
39- 70	* O1- .961	O1-1.011	F1-1.035	R1-1.075	T1-1.090
40- 71	* O1-1.039	J1-1.080	G1-1.096	Z1-1.109	C1-1.121
41- 72	* A1-1.018	J1-1.031	B1-1.113	Z1-1.122	W1-1.123
42- 73	* A1- .859	J1- .932	W1-1.056	Z1-1.103	T1-1.144
43- 74	* A1- .749	J1- .939	W1-1.027	T1-1.072	T1-1.081

```
46- 77  * X1- .8[illegible] W1- .[illegible] T1- .[illegible] V1- .[illegible] S1- .[illegible]
47- 78  * Y1- .699 X1- .755 T1- .736 W1- .[illegible] T1- .868
48- 79  * Y1- .[illegible] T1- .782 X1- .797 V1- .815 I1- .847
49- 80  * Y1- .376 V1- .[illegible] T1- .867 W1- .[illegible] I1- .894
50- 81  * Y1- .537 V1- .[illegible] W1- .879 F1- .[illegible] T1- .912
51- 82  * V1- .922 F1- .[illegible] E1- .911 Z1- .915 K1- .933
52- 83  * F1- .836 E1- .892 L1- .895 W1- .910 K1- .911
53- 84  * F1- .891 L1- .919 S1- .920 N1- .925 C1- .931
54- 85  * C1- .904 G1- .955 E1- .955 L1- .982 U1- .994
55- 86  * C1- .898 G1- .[illegible] C1- .942 H1-1.017 O1-1.023
56- 87  * G1- .890 O1- .905 C1- .921 H1-1.019 D1-1.035
57- 88  * O1- .922 G1- .929 C1- .935 D1-1.023 W1-1.038
58- 89  * O1- .993 G1-1.012 D1-1.017 J1-1.054 P1-1.090
59- 90  * D1-1.030 O1-1.060 C1-1.090 S1-1.112 B1-1.128
60- 91  * J1-1.072 D1-1.082 O1-1.087 T1-1.110 M1-1.158
61- 92  * J1- .973 T1-1.015 V1-1.126 I1-1.132 W1-1.137
62- 93  * J1- .914 T1- .959 I1-1.031 V1-1.035 M1-1.115
63- 94  * J1- .905 T1- .[illegible] I1- .982 V1-1.[illegible] M1-1.086
64- 95  * J1- .930 T1- .980 I1- .993 N1-1.047 V1-1.066
65- 96  * J1- .965 Z1- .974 Y1- .987 W1-1.[illegible] F1-1.004
66- 97  * Z1- .586 X1- .892 F1- .937 Y1- .945 M1- .962
67- 98  * Z1- .[illegible]31 X1- .[illegible] F1- .926 R1- .927 M1- .936
68- 99  * Z1- .[illegible]51 X1- .869 K1- .929 N1- .937 S1- .937
69-100  * Z1- .718 K1- .913 A1- .931 S1- .[illegible] W1- .969
70-101  * A1- .910 T1- .928 K1- .952 I1- .976 Z1-1.004
71-102  * I1- .921 A1- .934 T1- .938 K1-1.[illegible] J1-1.[illegible]
72-103  * I1- .[illegible] C1- .[illegible] L1-1.030 F1-1.[illegible] S1-1.[illegible]
73-104  * C1- .902 L1- .975 F1- .991 E1-1.035 U1-1.042
74-105  * C1- .[illegible]72 L1- .920 F1- .[illegible] D1- .[illegible] E1- .[illegible]
75-106  * L1- .890 U1- .[illegible] S1- .921 F1- .[illegible] S1- .[illegible]
76-107  * L1- .[illegible] U1- .919 F1- .974 P1-1.[illegible] E1-1.010
77-108  * U1- .984 L1- .987 S1-1.033 K1-1.[illegible] P1-1.046
78-109  * D1-1.[illegible] U1-1.[illegible] K1-1.[illegible] L1-1.107 M1-1.121
79-110  * D1-1.1[illegible] K1-1.19[illegible] U1-1.210 H1-1.21[illegible] I1-1.235
80-111  * D1-1.2[illegible] M1-1.257 N1-1.307 W1-1.307 K1-1.309
81-112  * M1-1.298 O1-1.351 W1-1.354 N1-1.380 U1-1.405
82-113  * M1-1.366 W1-1.393 D1-1.437 N1-1.447 Y1-1.[illegible]
83-114  * W1-1.424 M1-1.430 Y1-1.4[illegible] I1-1.[illegible] X1-1.461
84-115  * W1-1.[illegible] X1-1.48[illegible] M1-1.492 I1-1.[illegible] Y1-1.494
85-116  * W1-1.473 X1-1.[illegible] S1-1.[illegible] I1-1.[illegible] L1-1.499
86-117  * L1-1.487 W1-1.[illegible] X1-1.[illegible] F1-1.[illegible] S1-1.491
87-118  * J1-1.478 L1-1.[illegible] J1-1.[illegible] F1-1.[illegible] P1-1.[illegible]
88-119  * J1-1.[illegible] I1-1.[illegible] T1-1.[illegible] L1-1.[illegible] S1-1.[illegible]
89-120  * A1-1.000 E1-1.000 J1-1.000 Z1-1.000 B1-1.000
90-121  * A1-1.000 E1-1.000 J1-1.000 Z1-1.000 B1-1.000
91-122  * A1-1.000 E1-1.000 J1-1.000 Z1-1.000 B1-1.000
92-123  * A1-1.000 E1-1.000 J1-1.000 Z1-1.000 B1-1.000
93-124  * A1-1.000 E1-1.000 J1-1.000 Z1-1.000 B1-1.000
94-125  * A1-1.000 E1-1.000 J1-1.000 Z1-1.000 B1-1.000
95-126  * A1-1.000 E1-1.000 J1-1.000 Z1-1.000 B1-1.000
96-127  * A1-1.000 E1-1.000 J1-1.000 Z1-1.000 B1-1.000
97-128  * A1-1.000 E1-1.000 J1-1.000 Z1-1.000 B1-1.000
98-129  * A1-1.000 E1-1.000 J1-1.000 Z1-1.000 B1-1.000
99-130  * A1-1.000 E1-1.000 J1-1.000 Z1-1.000 B1-1.000
100-131 * A1-1.000 E1-1.000 J1-1.000 Z1-1.000 B1-1.000
101-132 * A1-1.000 E1-1.000 J1-1.000 Z1-1.000 B1-1.000
102-133 * A1-1.000 E1-1.000 J1-1.000 Z1-1.000 B1-1.000
```

RESULTS MATRIX SHOWING THE PROTOTYPE-TO-WINDOW
DISTANCES FOR PROTOTYPES MADE WITH VARIATION ENABLES
THE PROTOTYPE TO BE COMPARED TO WINDOWS OF WIDTH 32 COLUMNS
MATRIX IS SORTED BY DISTANCE WITHIN THE SENTENCE
COLUMNS. THE 1 TO 15 CLOSEST IDENTIFYING PROTOTYPES ARE
SHOWN FOR EACH WINDOW OF WIDTH 32 BEGINNING AND ENDING AT
THE SENTENCE COLUMN INDICATED BY THE LEFTMOST MATRIX COLUMN.

PROTOTYPES IN THE MATRIX ARE: A1 B1 C1 D1 E1 F1 G1 H1 I1 J1 K1 L1 M1 N1 O1 P1

Amplitude Spectrum – Sentence #1

MATRIX 1 OF 1

MATRIX OF THE 1 TO 5 CLOSEST IDENTIFYING PROTOTYPES FOR WINDOWS OF WIDTH 32

SENTENCE COLUMNS	GUESS 1 PROTO/DX	GUESS 2 PROTO/DX	GUESS 3 PROTO/DX	GUESS 4 PROTO/DX	GUESS 5 PROTO/DX
1- 32	A1- .302	D1- .[illegible]	F1- .[illegible]	N1- .[illegible]	P1- .[illegible]
2- 33	A1- .[illegible]	D1- .[illegible]	N1- .[illegible]	J1- .[illegible]	P1- .[illegible]
3- [illegible]	B1- .[illegible]	I1- .[illegible]	F1- .[illegible]	R1- .[illegible]	P1- .[illegible]
4- [illegible]	I1- .[illegible]	B1- .[illegible]	D1- .[illegible]	F1- .[illegible]	H1- .[illegible]
5- [illegible]	I1- .[illegible]	B1- .[illegible]	D1- .[illegible]	N1- .[illegible]	H1- .[illegible]
6- 37	I1- .[illegible]	F1- .[illegible]	N1- .[illegible]	H1- .[illegible]	
7- [illegible]	I1- .[illegible]	L1- .[illegible]	F1- .[illegible]	N1- .[illegible]	P1- .[illegible]
8- [illegible]	A1- .[illegible]	I1- .[illegible]	F1- .[illegible]	L1- .[illegible]	P1- .[illegible]
9- [illegible]	A1- .[illegible]	B1- .[illegible]	I1- .[illegible]	F1- .[illegible]	P1- .[illegible]
10- 41	A1- .[illegible]	B1- .[illegible]	L1- .[illegible]	R1- .[illegible]	I1- .[illegible]
11- [illegible]	A1- .[illegible]	B1- .[illegible]	F1- .[illegible]	L1- .[illegible]	P1- .[illegible]
12- [illegible]	A1- .[illegible]	B1- .[illegible]	F1- .[illegible]	L1- .[illegible]	P1- .[illegible]
13- [illegible]	A1- .[illegible]	B1- .[illegible]	F1- .[illegible]	G1- .[illegible]	K1- .[illegible]
14- [illegible]	A1- .[illegible]	B1- .[illegible]	F1- .[illegible]	R1- .[illegible]	K1- .[illegible]
15- [illegible]	A1- .[illegible]	K1- .[illegible]	I1- .[illegible]	B1- .[illegible]	P1- .[illegible]
16- [illegible]	A1- .[illegible]	K1- .[illegible]	A1- .[illegible]	L1- .[illegible]	P1- .[illegible]
17- [illegible]	K1- .[illegible]	A1- .[illegible]	G1- .[illegible]	N1- .[illegible]	P1- .[illegible]
18- 49	K1- .[illegible]	A1- .[illegible]	I1- .[illegible]	N1- .[illegible]	P1- .[illegible]
19- [illegible]	K1- .[illegible]	N1- .[illegible]	I1- .[illegible]	Y1- .[illegible]	M1- .[illegible]
20- 51	K1- .[illegible]	A1- .[illegible]	I1- .[illegible]	Y1- .[illegible]	M1- .[illegible]
21- [illegible]	K1- .[illegible]	I1- .[illegible]	Y1- .[illegible]	M1- .[illegible]	
22- 53	K1- .[illegible]	I1- .[illegible]	Y1- .[illegible]	M1- .[illegible]	
23- [illegible]	K1- .[illegible]	I1- .[illegible]	Y1- .[illegible]	M1- .[illegible]	
24- [illegible]	K1- .[illegible]	N1- .[illegible]	I1- .[illegible]	P1- .[illegible]	
25- [illegible]	K1- .[illegible]	G1- .[illegible]	Y1- .[illegible]	N1- .[illegible]	M1- .[illegible]
26- 57	G1- .[illegible]	K1- .[illegible]	Y1- .[illegible]	N1- .[illegible]	M1- .[illegible]
27- [illegible]	G1- .[illegible]	K1- .[illegible]	Y1- .[illegible]	P1- .[illegible]	
28- [illegible]	G1- .[illegible]	I1- .[illegible]	N1- .[illegible]	K1- .[illegible]	
29- [illegible]	G1- .[illegible]	I1- .[illegible]	N1- .[illegible]	P1- .[illegible]	
30- 61	G1- .[illegible]	I1- .[illegible]	N1- .[illegible]	P1- .[illegible]	
31- 62	G1- .[illegible]	I1- .[illegible]	N1- .[illegible]	P1- .[illegible]	
32- 63	G1- .[illegible]	I1- .[illegible]	N1- .[illegible]	P1- .[illegible]	
33- [illegible]	C1- .[illegible]	I1- .[illegible]	N1- .[illegible]	L1- .[illegible]	
34- [illegible]	L1- .[illegible]	I1- .[illegible]	E1- .[illegible]	J1- .[illegible]	
35- [illegible]	L1- .[illegible]	I1- .[illegible]	J1- .[illegible]	E1- .[illegible]	
36- [illegible]	L1- .[illegible]	I1- .[illegible]	J1- .[illegible]	F1- .[illegible]	
37- [illegible]	I1- .[illegible]	L1- .[illegible]	J1- .[illegible]	F1- .[illegible]	
38- [illegible]	I1- .[illegible]	L1- .[illegible]	J1- .[illegible]	F1- .[illegible]	
39- [illegible]	I1- .[illegible]	L1- .[illegible]	J1- .[illegible]	M1- .[illegible]	
40- [illegible]	I1- .[illegible]	L1- .[illegible]	J1- .[illegible]	M1- .[illegible]	
41- [illegible]	I1- .[illegible]	L1- .[illegible]	J1- .[illegible]	M1- .[illegible]	
42- [illegible]	I1- .[illegible]	L1- .[illegible]	J1- .[illegible]	M1- .[illegible]	
43- [illegible]	I1- .[illegible]	L1- .[illegible]	J1- .[illegible]	M1- .[illegible]	

Ran = 1.61

Ran = 1.2

45- 76	*	D1-	.28[illegible]	T1-	.31[illegible]	D1-	.329	I1-	.3[illegible]	C1-	.372
46- 77	*	D1-	.26[illegible]	U1-	.[illegible]	C1-	.3[illegible]	S1-	.3[illegible]	T1-	.393
47- 78	*	D1-	.27[illegible]	D1-	.33[illegible]	C1-	.3[illegible]	J1-	.3[illegible]	C1-	.3[illegible]
48- 79	*	D1-	.3[illegible]	D1-	.33[illegible]	C1-	.347	D1-	.3[illegible]	M1-	.365
49- 80	*	D1-	.32[illegible]	F1-	.33[illegible]	C1-	.3[illegible]	C1-	.3[illegible]	C1-	.379
50- 81	*	D1-	.342	C1-	.3[illegible]	C1-	.3[illegible]	Y1-	.313	F1-	.386
51- 82	*	Y1-	.349	A1-	.371	C1-	.372	K1-	.3[illegible]	M1-	.393
52- 83	*	A1-	.33[illegible]	Y1-	.3[illegible]	K1-	.3[illegible]	A1-	.3[illegible]	M1-	.400
53- 84	*	A1-	.33[illegible]	Y1-	.369	K1-	.396	M1-	.41[illegible]	V1-	.426
54- 85	*	A1-	.333	K1-	.3[illegible]	Y1-	.39[illegible]	M1-	.42[illegible]	V1-	.439
55- 86	*	K1-	.33[illegible]	A1-	.33[illegible]	Y1-	.411	M1-	.418	F1-	.43[illegible]
56- 87	*	K1-	.31[illegible]	A1-	.3[illegible]	F1-	.40[illegible]	M1-	.41[illegible]	Y1-	.431
57- 88	*	K1-	.297	F1-	.373	A1-	.373	M1-	.410	F1-	.412
58- 89	*	K1-	.28[illegible]	F1-	.3[illegible]	F1-	.33[illegible]	A1-	.39[illegible]	D1-	.398
59- 90	*	K1-	.283	F1-	.321	F1-	.359	P1-	.37[illegible]	J1-	.393
60- 91	*	K1-	.25[illegible]	F1-	.30[illegible]	F1-	.337	P1-	.3[illegible]	T1-	.387
61- 92	*	F1-	.30[illegible]	K1-	.302	F1-	.319	P1-	.3[illegible]	R1-	.37[illegible]
62- 93	*	F1-	.31[illegible]	F1-	.31[illegible]	K1-	.3[illegible]	D1-	.3[illegible]	P1-	.380
63- 94	*	F1-	.32[illegible]	F1-	.3[illegible]	F1-	.3[illegible]	K1-	.3[illegible]	T1-	.407
64- 95	*	E1-	.357	F1-	.3[illegible]	F1-	.3[illegible]	K1-	.41[illegible]	C1-	.42[illegible]
65- 96	*	F1-	.403	B1-	.4[illegible]	S1-	.4[illegible]	F1-	.[illegible]	K1-	.473
66- 97	*	F1-	.453	B1-	.[illegible]	P1-	.49[illegible]	F1-	.5[illegible]	F1-	.527
67- 98	*	E1-	.497	S1-	.[illegible]	F1-	.542	B1-	.5[illegible]	F1-	.55[illegible]
68- 99	*	F1-	.477	F1-	.[illegible]21	C1-	.52[illegible]	T1-	.[illegible]	D1-	.[illegible]32
69-100	*	E1-	.422	T1-	.[illegible]1	F1-	.4[illegible]	B1-	.[illegible]	A1-	.450
70-101	*	E1-	.37[illegible]	T1-	.37[illegible]	F1-	.402	R1-	.42[illegible]	C1-	.431
71-102	*	T1-	.318	F1-	.3[illegible]	F1-	.3[illegible]7	R1-	.3[illegible]	T1-	.400
72-103	*	T1-	.283	F1-	.323	F1-	.33[illegible]	I1-	.333	J1-	.360
73-104	*	T1-	.267	F1-	.297	F1-	.318	T1-	.3[illegible]	J1-	.330
74-105	*	F1-	.266	T1-	.20[illegible]	F1-	.307	I1-	.31[illegible]	J1-	.314
75-106	*	F1-	.2[illegible]	T1-	.2[illegible]	F1-	.310	J1-	.312	T1-	.312
76-107	*	F1-	.238	F1-	.302	T1-	.311	J1-	.32[illegible]	D1-	.332
77-108	*	F1-	.249	F1-	.31[illegible]	P1-	.33[illegible]	P1-	.33[illegible]	T1-	.351
78-109	*	F1-	.261	B1-	.332	F1-	.341	P1-	.337	J1-	.399
79-110	*	F1-	.329	B1-	.336	F1-	.37[illegible]	P1-	.33[illegible]	F1-	.416
80-111	*	B1-	.34[illegible]	F1-	.3[illegible]	F1-	.403	E1-	.41[illegible]	B1-	.432
81-112	*	B1-	.37[illegible]	P1-	.440	F1-	.494	D1-	.43[illegible]	C1-	.464
82-113	*	B1-	.409	P1-	.4[illegible]	F1-	.47[illegible]	D1-	.4[illegible]	C1-	.481
83-114	*	B1-	.422	P1-	.471	C1-	.473	R1-	.47[illegible]	C1-	.483
84-115	*	B1-	.439	C1-	.4[illegible]	Y1-	.472	D1-	.47[illegible]	F1-	.479
85-116	*	C1-	.4[illegible]	B1-	.4[illegible]2	Y1-	.4[illegible]	D1-	.4[illegible]	C1-	.47[illegible]
86-117	*	C1-	.4[illegible]1	G1-	.4[illegible]	Y1-	.47[illegible]	D1-	.[illegible]1	T1-	.480
87-118	*	C1-	.43[illegible]	G1-	.4[illegible]	C1-	.4[illegible]	Y1-	.4[illegible]	T1-	.48[illegible]
88-119	*	C1-	.423	G1-	.[illegible]32	C1-	.4[illegible]	T1-	.[illegible]11	A1-	.493
89-120	*	G1-	.408	C1-	.[illegible]21	C1-	.4[illegible]	D1-	.[illegible]	C1-	.497
90-121	*	G1-	.39[illegible]	C1-	.[illegible]32	C1-	.4[illegible]3	D1-	.[illegible]	C1-	.501
91-122	*	G1-	.389	C1-	.[illegible]	C1-	.463	D1-	.[illegible]	C1-	.511
92-123	*	G1-	.38[illegible]	C1-	.[illegible]51	C1-	.40[illegible]	D1-	.[illegible]	C1-	.51[illegible]
93-124	*	G1-	.373	C1-	.[illegible]73	C1-	.463	D1-	.[illegible]	C1-	.498
94-125	*	G1-	.363	C1-	.4[illegible]	C1-	.[illegible]	D1-	.4[illegible]	C1-	.47[illegible]
95-126	*	G1-	.36[illegible]	C1-	.39[illegible]	C1-	.451	D1-	.[illegible]32	C1-	.462
96-127	*	G1-	.37[illegible]	C1-	.3[illegible]	D1-	.4[illegible]7	F1-	.[illegible]	C1-	.4[illegible]
97-128	*	C1-	.361	G1-	.3[illegible]	F1-	.428	D1-	.[illegible]53	C1-	.4[illegible]2
98-129	*	C1-	.38[illegible]	G1-	.4[illegible]	F1-	.41[illegible]	T1-	.[illegible]39	D1-	.441
99-130	*	C1-	.382	F1-	.4[illegible]	C1-	.412	T1-	.[illegible]31	D1-	.432
100-131	*	C1-	.38[illegible]	F1-	.4[illegible]	D1-	.41[illegible]	G1-	.423	T1-	.42[illegible]
101-132	*	C1-	.391	F1-	.4[illegible]	D1-	.407	T1-	.4[illegible]	E1-	.432
102-133	*	C1-	.39[illegible]	T1-	.3[illegible]	F1-	.4[illegible]	T1-	.[illegible]1[illegible]	T1-	.438

RESULTS MATRIX SHOWING THE PROTOTYPE-TO-WINDOW
DISTANCES FOR PROTOTYPES WHOSE WIDTH VALIDATION ENABLES
THE PROTOTYPE TO BE COMPARED TO WINDOWS OF WIDTH 32 COLUMNS
MATRIX IS SORTED BY DISTANCE WITHIN THE SENTENCE
COLUMNS. THE 1 TO 13 CLOSEST IDENTIFYING PROTOTYPES ARE
SHOWN FOR EACH WINDOW OF WIDTH 32 BEGINNING AND ENDING AT
THE SENTENCE COLUMN INDICATED BY THE LEFTMOST MATRIX COLUMN.

PROTOTYPES IN THE MATRIX ARE: A1 B1 C1 D1 E1 F1 G1 H1 I1 J1 K1 L1 M1 N1 O1 P1

Amplitude Spectrum – Sentence #2

MATRIX 1 OF 1

MATRIX OF THE 1 TO 13 CLOSEST IDENTIFYING PROTOTYPES FOR WINDOWS OF WIDTH 32

SENTENCE COLUMNS	GUESS 1 PROTO/DX	GUESS 2 PROTO/DX	GUESS 3 PROTO/DX	GUESS 4 PROTO/DX
1- 32	T1- .[illegible]	T1- .[illegible]	T1- .371	I1- .[illegible]
2- 33	T1- .[illegible]	I1- .[illegible]	D1- .[illegible]	I1- .[illegible]
3- 34	T1- .[illegible]	T1- .[illegible]	T1- .[illegible]	I1- .[illegible]
4- 35	T1- .[illegible]	D1- .[illegible]	T1- .381	I1- .[illegible]
5- 36	T1- .371	D1- .[illegible]	T1- .391	E1- .[illegible]
6- 37	D1- .[illegible]	T1- .[illegible]	T1- .[illegible]	E1- .[illegible]
7- 38	D1- .[illegible]	T1- .[illegible]	I1- .[illegible]	E1- .[illegible]
8- 39	T1- .[illegible]	D1- .[illegible]	T1- .[illegible]	E1- .[illegible]
9- 40	T1- .[illegible]	D1- .[illegible]	T1- .411	I1- .[illegible]
10- 41	T1- .[illegible]	D1- .[illegible]	T1- .[illegible]	I1- .[illegible]
11- 42	T1- .[illegible]	I1- .[illegible]	L1- .[illegible]	H1- .[illegible]
12- 43	T1- .[illegible]	L1- .[illegible]	D1- .[illegible]	H1- .[illegible]
13- 44	T1- .[illegible]	H1- .[illegible]	T1- .[illegible]	L1- .[illegible]
14- 45	D1- .[illegible]	H1- .[illegible]	T1- .[illegible]	D1- .[illegible]
15- 46	D1- .[illegible]	H1- .[illegible]	T1- .[illegible]	D1- .[illegible]
16- 47	D1- .[illegible]	H1- .[illegible]	T1- .[illegible]	D1- .[illegible]
17- 48	D1- .[illegible]	T1- .[illegible]	H1- .[illegible]	D1- .[illegible]
18- 49	T1- .[illegible]	D1- .[illegible]	J1- .[illegible]	D1- .[illegible]
19- 50	T1- .[illegible]	H1- .[illegible]	D1- .[illegible]	D1- .[illegible]
20- 51	T1- .[illegible]	I1- .[illegible]	J1- .[illegible]	D1- .[illegible]
21- 52	T1- .[illegible]	I1- .[illegible]	J1- .[illegible]	D1- .[illegible]
22- 53	T1- .[illegible]	I1- .[illegible]	H1- .[illegible]	D1- .[illegible]
23- 54	I1- .[illegible]	I1- .[illegible]	T1- .[illegible]	D1- .[illegible]
24- 55	T1- .[illegible]	D1- .[illegible]	I1- .[illegible]	T1- .[illegible]
25- 56	D1- .[illegible]	T1- .[illegible]	I1- .[illegible]	I1- .[illegible]
26- 57	D1- .[illegible]	I1- .[illegible]	T1- .[illegible]	D1- .[illegible]
27- 58	D1- .[illegible]	D1- .[illegible]	T1- .[illegible]	D1- .[illegible]
28- 59	D1- .[illegible]	I1- .[illegible]	T1- .[illegible]	D1- .[illegible]
29- 60	D1- .[illegible]	T1- .[illegible]	I1- .[illegible]	D1- .[illegible]
30- 61	D1- .[illegible]	D1- .[illegible]	T1- .[illegible]	I1- .[illegible]
31- 62	D1- .[illegible]	D1- .[illegible]	T1- .[illegible]	K1- .[illegible]
32- 63	D1- .[illegible]	K1- .[illegible]	T1- .[illegible]	I1- .[illegible]
33- 64	K1- .[illegible]	D1- .[illegible]	T1- .[illegible]	I1- .[illegible]
34- 65	K1- .[illegible]	D1- .[illegible]	T1- .[illegible]	I1- .[illegible]
35- 66	K1- .[illegible]	D1- .[illegible]	D1- .[illegible]	K1- .[illegible]
36- 67	K1- .[illegible]	K1- .[illegible]	T1- .[illegible]	I1- .[illegible]
37- 68	K1- .[illegible]	K1- .[illegible]	T1- .[illegible]	I1- .[illegible]
38- 69	K1- .[illegible]	K1- .[illegible]	T1- .[illegible]	D1- .[illegible]
39- 70	K1- .[illegible]	K1- .[illegible]	T1- .[illegible]	D1- .[illegible]
40- 71	D1- .[illegible]	D1- .[illegible]	T1- .[illegible]	K1- .[illegible]
41- 72	D1- .[illegible]	D1- .[illegible]	T1- .[illegible]	D1- .[illegible]
42- 73	T1- .[illegible]	D1- .[illegible]	T1- .[illegible]	D1- .[illegible]
43- 74	T1- .[illegible]	D1- .[illegible]	T1- .[illegible]	D1- .[illegible]

```
45- 75  * T1- .297  Y1- .271  F1- .304  B1- .[illegible]  J1- .423
46- 77  * T1- .[illegible]  Y1- .[illegible]  F1- .[illegible]  J1- .[illegible]  P1- .[illegible]
47- 79  * T1- .[illegible]  I1- .[illegible]  F1- .[illegible]  J1- .[illegible]  P1- .[illegible]
48- 81  * T1- .[illegible]  Y1- .[illegible]  F1- .[illegible]  J1- .[illegible]  P1- .411
49- 83  * T1- .[illegible]  I1- .[illegible]  F1- .[illegible]  J1- .[illegible]  P1- .[illegible]
50- 85  * T1- .[illegible]  B1- .[illegible]  T1- .[illegible]  B1- .[illegible]  F1- .[illegible]
51- 87  * T1- .[illegible]  B1- .[illegible]  F1- .[illegible]  B1- .[illegible]  F1- .436
52- 89  * B1- .[illegible]  P1- .[illegible]  F1- .[illegible]  T1- .[illegible]  F1- .[illegible]
53- 91  * B1- .[illegible]  P1- .[illegible]  B1- .[illegible]  K1- .[illegible]  F1- .491
54- 93  * N1- .[illegible]  P1- .[illegible]  Y1- .[illegible]  M1- .[illegible]  M1- .[illegible]
55- 95  * N1- .[illegible]  X1- .[illegible]  M1- .[illegible]  B1- .[illegible]  Y1- .479
56- 97  * N1- .[illegible]  K1- .[illegible]  V1- .[illegible]  H1- .[illegible]  Y1- .[illegible]
57- 99  * N1- .[illegible]  X1- .[illegible]  V1- .[illegible]  H1- .[illegible]  Y1- .[illegible]
58- 101 * N1- .[illegible]  X1- .[illegible]  M1- .[illegible]  V1- .[illegible]  Y1- .[illegible]
59- 103 * X1- .[illegible]  M1- .[illegible]  Y1- .[illegible]  V1- .[illegible]  Y1- .449
60- 105 * N1- .[illegible]  H1- .[illegible]  Y1- .[illegible]  V1- .[illegible]  Y1- .[illegible]
61- 107 * N1- .[illegible]  M1- .[illegible]  V1- .[illegible]  X1- .[illegible]  Y1- .[illegible]
62- 109 * M1- .[illegible]  N1- .[illegible]  V1- .[illegible]  X1- .[illegible]  Y1- .[illegible]
63- 111 * M1- .[illegible]  N1- .[illegible]  V1- .[illegible]  K1- .[illegible]  Y1- .489
64- 113 * M1- .[illegible]  V1- .[illegible]  T1- .[illegible]  X1- .[illegible]  F1- .[illegible]
65- 115 * M1- .[illegible]  V1- .[illegible]  B1- .[illegible]  X1- .[illegible]  P1- .[illegible]
66- 117 * M1- .[illegible]  B1- .[illegible]  F1- .[illegible]  X1- .[illegible]  M1- .[illegible]
67- 119 * M1- .[illegible]  P1- .[illegible]  F1- .[illegible]  I1- .[illegible]  T1- .[illegible]
68- 121 * P1- .[illegible]  T1- .[illegible]  T1- .[illegible]  B1- .[illegible]  J1- .[illegible]
69-122  * T1- .[illegible]  T1- .[illegible]  F1- .[illegible]  J1- .[illegible]  P1- .[illegible]
70-123  * T1- .[illegible]  T1- .[illegible]  F1- .[illegible]  J1- .[illegible]  P1- .412
71-124  * T1- .[illegible]  P1- .[illegible]  B1- .[illegible]  J1- .[illegible]  P1- .396
72-125  * T1- .[illegible]  P1- .[illegible]  F1- .[illegible]  I1- .[illegible]  J1- .[illegible]
73-126  * T1- .[illegible]  P1- .[illegible]  P1- .[illegible]  K1- .[illegible]  F1- .[illegible]
74-127  * R1- .[illegible]  P1- .[illegible]  F1- .[illegible]  T1- .[illegible]  F1- .424
75-128  * B1- .[illegible]  B1- .[illegible]  F1- .[illegible]  K1- .[illegible]  T1- .457
76-129  * B1- .[illegible]  C1- .[illegible]  M1- .[illegible]  B1- .[illegible]  I1- .[illegible]
77-130  * K1- .[illegible]  N1- .[illegible]  B1- .[illegible]  K1- .[illegible]  P1- .[illegible]
78-131  * N1- .[illegible]  X1- .[illegible]  F1- .[illegible]  K1- .[illegible]  M1- .[illegible]
79-132  * B1- .[illegible]  X1- .[illegible]  K1- .[illegible]  B1- .[illegible]  M1- .[illegible]
80-133  * N1- .[illegible]  X1- .[illegible]  B1- .[illegible]  N1- .[illegible]  F1- .[illegible]
81-134  * N1- .[illegible]  X1- .[illegible]  T1- .[illegible]  K1- .[illegible]  Y1- .[illegible]
82-135  * N1- .[illegible]  X1- .[illegible]  T1- .[illegible]  K1- .[illegible]  Y1- .[illegible]
83-136  * N1- .[illegible]  Y1- .[illegible]  M1- .[illegible]  K1- .[illegible]  Y1- .[illegible]
84-137  * B1- .[illegible]  X1- .[illegible]  M1- .[illegible]  K1- .[illegible]  Y1- .[illegible]
85-138  * N1- .[illegible]  X1- .[illegible]  T1- .[illegible]  K1- .[illegible]  Y1- .[illegible]
86-139  * N1- .[illegible]  Y1- .[illegible]  T1- .[illegible]  K1- .[illegible]  T1- .[illegible]
87-140  * N1- .[illegible]  X1- .[illegible]  T1- .[illegible]  K1- .[illegible]  T1- .[illegible]
88-141  * N1- .[illegible]  X1- .[illegible]  T1- .[illegible]  K1- .[illegible]  M1- .[illegible]
89-142  * N1- .[illegible]  N1- .[illegible]  M1- .[illegible]  X1- .[illegible]  M1- .[illegible]
90-143  * Y1- .[illegible]  K1- .[illegible]  M1- .[illegible]  H1- .[illegible]  Y1- .[illegible]
91-144  * N1- .[illegible]  K1- .[illegible]  T1- .[illegible]  N1- .[illegible]  Y1- .[illegible]
92-145  * N1- .[illegible]  N1- .[illegible]  K1- .[illegible]  Y1- .[illegible]  M1- .[illegible]
93-146  * N1- .[illegible]  M1- .[illegible]  K1- .[illegible]  Y1- .[illegible]  M1- .[illegible]
94-147  * N1- .[illegible]  M1- .[illegible]  Y1- .[illegible]  K1- .[illegible]  M1- .[illegible]
95-148  * N1- .[illegible]  Y1- .[illegible]  T1- .[illegible]  T1- .[illegible]  M1- .[illegible]
96-149  * Y1- .[illegible]  N1- .[illegible]  M1- .[illegible]  C1- .[illegible]  M1- .[illegible]
97-150  * B1- .[illegible]  Y1- .[illegible]  M1- .[illegible]  N1- .[illegible]  T1- .[illegible]
98-151  * C1- .[illegible]  C1- .[illegible]  C1- .[illegible]  Y1- .[illegible]  C1- .423
99-152  * C1- .[illegible]  C1- .[illegible]  C1- .[illegible]  C1- .[illegible]  C1- .424
100-153 * C1- .[illegible]  C1- .[illegible]  C1- .[illegible]  C1- .[illegible]  C1- .[illegible]
101-154 * C1- .[illegible]  C1- .[illegible]  C1- .[illegible]  C1- .[illegible]  C1- .[illegible]
102-155 * C1- .[illegible]  C1- .[illegible]  C1- .[illegible]  C1- .[illegible]  C1- .[illegible]
```

RESULTS MATRIX SHOWING THE PROTOTYPE-TO-WINDOW
DISTANCES FOR PROTOTYPES WHOSE WIDTH VARIATION ENABLES
THE PROTOTYPE TO BE COMPARED TO WINDOWS OF WIDTH 32 COLUMNS
MATRIX IS SORTED BY DISTANCE WITHIN THE SENTENCE
COLUMNS. THE 1 TO 13 CLOSEST IDENTIFYING PROTOTYPES ARE
SHOWN FOR EACH WINDOW OF WIDTH 32 BEGINNING AND ENDING AT
THE SENTENCE COLUMN INDICATED BY THE LEFTMOST MATRIX COLUMN.

PROTOTYPES IN THE MATRIX ARE: A1 F1 C1 D1 T1 F1 G1 H1 I1 J1 K1 L1 M1 N1 O1 P1

Amplitude Spectrum - Sentence # 3

MATRIX 1 OF 1

MATRIX OF THE 1 TO 3 CLOSEST IDENTIFYING PROTOTYPES FOR WINDOWS OF WIDTH 32

SENTENCE COLUMN	GUESS 1 PROTOTYPE	GUESS 2 PROTOTYPE	GUESS 3 PROTOTYPE	GUESS 4 PROTOTYPE
1- 32	W1- .363	Y1- .384	A1- .389	H1- .[illegible]	T1- .[illegible]
2- 33	W1- .[illegible]	Y1- .[illegible]	A1- .[illegible]	H1- .[illegible]	Y1- .[illegible]
3- 34	W1- .[illegible]	Y1- .[illegible]	T1- .412	K1- .[illegible]	W1- .[illegible]
4- 35	W1- .[illegible]	Y1- .[illegible]	A1- .[illegible]	O1- .[illegible]	T1- .[illegible]
5- 36	W1- .[illegible]	Y1- .[illegible]	T1- .[illegible]	O1- .[illegible]	T1- .[illegible]
6- 37	O1- .[illegible]	W1- .[illegible]	Y1- .[illegible]	O1- .[illegible]	T1- .[illegible]
7- 38	O1- .[illegible]	T1- .[illegible]	W1- .[illegible]	Y1- .[illegible]	T1- .[illegible]
8- 39	O1- .[illegible]	T1- .[illegible]	T1- .[illegible]	O1- .[illegible]	W1- .[illegible]
9- 40	O1- .[illegible]	T1- .[illegible]	T1- .[illegible]	O1- .[illegible]	W1- .[illegible]
10- 41	O1- .413	T1- .413	T1- .[illegible]	O1- .[illegible]	W1- .[illegible]
11- 42	O1- .[illegible]	O1- .[illegible]	T1- .[illegible]	W1- .[illegible]	T1- .[illegible]
12- 43	O1- .[illegible]	O1- .[illegible]	T1- .[illegible]	W1- .[illegible]	Y1- .[illegible]
13- 44	O1- .[illegible]	T1- .[illegible]	K1- .[illegible]	A1- .[illegible]	Y1- .[illegible]
14- 45	T1- .[illegible]	K1- .[illegible]	A1- .[illegible]	O1- .[illegible]	T1- .[illegible]
15- 46	K1- .[illegible]	T1- .[illegible]	A1- .[illegible]	T1- .[illegible]	O1- .[illegible]
16- 47	K1- .[illegible]	T1- .[illegible]	F1- .[illegible]	A1- .[illegible]	T1- .[illegible]
17- 48	K1- .[illegible]	T1- .[illegible]	T1- .[illegible]	K1- .[illegible]	T1- .[illegible]
18- 49	K1- .[illegible]	T1- .[illegible]	T1- .[illegible]	O1- .[illegible]	T1- .[illegible]
19- 50	K1- .[illegible]	T1- .[illegible]	T1- .[illegible]	T1- .[illegible]	T1- .[illegible]
20- 51	K1- .[illegible]	T1- .[illegible]	T1- .[illegible]	A1- .[illegible]	T1- .[illegible]
21- 52	A1- .[illegible]	A1- .[illegible]	K1- .[illegible]	T1- .[illegible]	T1- .[illegible]
22- 53	A1- .[illegible]	A1- .[illegible]	T1- .[illegible]	A1- .[illegible]	T1- .[illegible]
23- 54	A1- .[illegible]	A1- .[illegible]	T1- .[illegible]	T1- .[illegible]	W1- .[illegible]
24- 55	A1- .[illegible]	A1- .[illegible]	T1- .[illegible]	K1- .[illegible]	T1- .[illegible]
25- 56	A1- .[illegible]	T1- .[illegible]	T1- .[illegible]	K1- .[illegible]	T1- .[illegible]
26- 57	A1- .[illegible]	A1- .[illegible]	T1- .[illegible]	K1- .[illegible]	T1- .[illegible]
27- 58	A1- .[illegible]	T1- .[illegible]	W1- .[illegible]	A1- .[illegible]	T1- .[illegible]
28- 59	A1- .[illegible]	A1- .[illegible]	W1- .[illegible]	A1- .[illegible]	W1- .[illegible]
29- 60	A1- .[illegible]	A1- .[illegible]	W1- .[illegible]	A1- .[illegible]	W1- .[illegible]
30- 61	A1- .[illegible]	A1- .[illegible]	W1- .[illegible]	T1- .[illegible]	T1- .[illegible]
31- 62	A1- .[illegible]	A1- .[illegible]	A1- .[illegible]	K1- .[illegible]	T1- .[illegible]
32- 63	A1- .[illegible]	A1- .[illegible]	A1- .[illegible]	K1- .[illegible]	A1- .[illegible]
33- 64	A1- .[illegible]	A1- .[illegible]	A1- .[illegible]	K1- .[illegible]	W1- .[illegible]
34- 65	A1- .[illegible]	A1- .[illegible]	A1- .[illegible]	K1- .[illegible]	W1- .[illegible]
35- 66	A1- .[illegible]	A1- .[illegible]	A1- .[illegible]	K1- .[illegible]	T1- .[illegible]
36- 67	A1- .[illegible]	A1- .[illegible]	A1- .[illegible]	K1- .[illegible]	T1- .[illegible]
37- 68	A1- .[illegible]	A1- .[illegible]	A1- .[illegible]	K1- .[illegible]	T1- .[illegible]
38- 69	A1- .[illegible]	W1- .[illegible]	A1- .[illegible]	T1- .[illegible]	T1- .[illegible]
39- 70	A1- .[illegible]	W1- .[illegible]	A1- .[illegible]	T1- .[illegible]	W1- .[illegible]
40- 71	A1- .[illegible]	T1- .[illegible]	W1- .[illegible]	A1- .[illegible]	T1- .[illegible]
41- 72	A1- .[illegible]	T1- .[illegible]	W1- .[illegible]	A1- .[illegible]	T1- .[illegible]
42- 73	A1- .[illegible]	T1- .[illegible]	W1- .[illegible]	A1- .[illegible]	T1- .[illegible]
43- 74	A1- .[illegible]	T1- .[illegible]	T1- .[illegible]	K1- .[illegible]	T1- .[illegible]

RESULTS MATRIX SHOWING THE PROTOTYPE-TO-WINDOW
DISTANCES FOR PROTOTYPES WHOSE WIDTH VARIATION ENABLES
THE PROTOTYPE TO BE COMPARED TO WINDOWS OF WIDTH 32 COLUMNS
MATRIX IS SORTED BY DISTANCE WITHIN THE SENTENCE
COLUMNS. THE 1 TO 13 CLOSEST IDENTIFYING PROTOTYPES ARE
SHOWN FOR EACH WINDOW OF WIDTH 32 BEGINNING AND ENDING AT
THE SENTENCE COLUMN INDICATED BY THE LEFTMOST MATRIX COLUMN.

PROTOTYPES IN THE MATRIX ARE: A1 B1 C1 D1 E1 F1 G1 H1 I1 J1 K1 L1 M1 N1 O1 P1

Amplitude Spectrum — Sentence # 24

MATRIX 1 OF 1

MATRIX OF THE 1 TO 5 CLOSEST IDENTIFYING PROTOTYPES FOR WINDOWS OF WIDTH 32

SENTENCE COLUMNS	GUESS 1 PROTO/DX	GUESS 2 PROTO/DX	GUESS 3 PROTO/DX	GUESS 4 PROTO/DX	GUESS 5 PROTO/DX
1- 32	V1- .376	Y1- .347	A1- .428	O1- .[illegible]	P1- .521
2- 33	V1- .311	Y1- .[illegible]	A1- .[illegible]	O1- .[illegible]	P1- .[illegible]
3- 34	V1- .[illegible]	Y1- .[illegible]	A1- .[illegible]	M1- .[illegible]	O1- .[illegible]
4- 35	V1- .[illegible]	Y1- .[illegible]	A1- .[illegible]	M1- .[illegible]	O1- .[illegible]
5- 36	V1- .[illegible]	Y1- .[illegible]	A1- .[illegible]	M1- .[illegible]	O1- .[illegible]
6- 37	V1- .[illegible]	Y1- .[illegible]	A1- .[illegible]	M1- .[illegible]	M1- .[illegible]
7- 38	V1- .[illegible]	Y1- .[illegible]	A1- .[illegible]	M1- .[illegible]	A1- .[illegible]
8- 39	V1- .[illegible]	Y1- .[illegible]	A1- .[illegible]	M1- .[illegible]	P1- .[illegible]
9- 40	V1- .[illegible]	Y1- .[illegible]	A1- .[illegible]	M1- .[illegible]	P1- .[illegible]
10- 41	V1- .[illegible]	Y1- .[illegible]	A1- .[illegible]	M1- .[illegible]	P1- .[illegible]
11- 42	Y1- .[illegible]	V1- .[illegible]	A1- .[illegible]	M1- .[illegible]	P1- .[illegible]
12- 43	Y1- .[illegible]	V1- .[illegible]	A1- .[illegible]	O1- .[illegible]	P1- .[illegible]
13- 44	Y1- .[illegible]	A1- .[illegible]	O1- .[illegible]	M1- .[illegible]	P1- .[illegible]
14- 45	Y1- .[illegible]	O1- .[illegible]	V1- .[illegible]	P1- .[illegible]	P1- .[illegible]
15- 46	P1- .[illegible]	Y1- .[illegible]	O1- .[illegible]	P1- .[illegible]	V1- .[illegible]
16- 47	O1- .[illegible]	Y1- .[illegible]	O1- .[illegible]	P1- .[illegible]	A1- .[illegible]
17- 48	O1- .[illegible]	Y1- .[illegible]	O1- .[illegible]	P1- .[illegible]	O1- .[illegible]
18- 49	O1- .[illegible]	Y1- .[illegible]	O1- .[illegible]	P1- .[illegible]	M1- .[illegible]
19- 50	O1- .[illegible]	M1- .[illegible]	Y1- .[illegible]	M1- .[illegible]	M1- .[illegible]
20- 51	Y1- .[illegible]	M1- .[illegible]	Y1- .[illegible]	O1- .[illegible]	M1- .[illegible]
21- 52	Y1- .[illegible]	M1- .[illegible]	Y1- .[illegible]	K1- .[illegible]	A1- .[illegible]
22- 53	Y1- .721	M1- .[illegible]	Y1- .[illegible]	K1- .[illegible]	M1- .[illegible]
23- 54	Y1- .[illegible]	M1- .[illegible]	Y1- .[illegible]	M1- .[illegible]	M1- .[illegible]
24- 55	Y1- .[illegible]	M1- .[illegible]	A1- .[illegible]	Y1- .[illegible]	M1- .[illegible]
25- 56	K1- .[illegible]	M1- .[illegible]	A1- .[illegible]	Y1- .[illegible]	M1- .[illegible]
26- 57	Y1- .[illegible]	M1- .[illegible]	M1- .[illegible]	Y1- .[illegible]	M1- .[illegible]
27- 58	Y1- .[illegible]	M1- .[illegible]	A1- .[illegible]	Y1- .[illegible]	M1- .[illegible]
28- 59	K1- .[illegible]	M1- .[illegible]	A1- .[illegible]	Y1- .[illegible]	M1- .[illegible]
29- 60	Y1- .[illegible]	M1- .[illegible]	K1- .[illegible]	Y1- .[illegible]	A1- .[illegible]
30- 61	Y1- .[illegible]	M1- .[illegible]	K1- .[illegible]	Y1- .[illegible]	A1- .[illegible]
31- 62	Y1- .[illegible]	M1- .[illegible]	K1- .[illegible]	O1- .[illegible]	Y1- .[illegible]
32- 63	Y1- .[illegible]	M1- .[illegible]	K1- .[illegible]	O1- .[illegible]	O1- .[illegible]
33- 64	K1- .[illegible]	O1- .[illegible]	K1- .[illegible]	M1- .[illegible]	O1- .[illegible]
34- 65	O1- .[illegible]	K1- .[illegible]	O1- .[illegible]	K1- .[illegible]	M1- .[illegible]
35- 66	O1- .[illegible]	O1- .[illegible]	K1- .[illegible]	X1- .[illegible]	O1- .[illegible]
36- 67	O1- .[illegible]	O1- .[illegible]	O1- .[illegible]	O1- .[illegible]	M1- .[illegible]
37- 68	O1- .[illegible]	O1- .[illegible]	O1- .[illegible]	A1- .[illegible]	O1- .[illegible]
38- 69	O1- .[illegible]	O1- .[illegible]	O1- .[illegible]	Y1- .[illegible]	O1- .[illegible]
39- 70	Y1- .[illegible]	O1- .[illegible]	V1- .[illegible]	O1- .[illegible]	A1- .[illegible]
40- 71	Y1- .[illegible]	V1- .[illegible]	A1- .[illegible]	O1- .[illegible]	O1- .[illegible]
41- 72	Y1- .[illegible]	A1- .[illegible]	A1- .[illegible]	O1- .[illegible]	O1- .[illegible]
42- 73	Y1- .[illegible]	M1- .[illegible]	A1- .[illegible]	A1- .[illegible]	M1- .[illegible]
43- 74	Y1- .[illegible]	M1- .[illegible]	A1- .[illegible]	A1- .[illegible]	A1- .[illegible]

```
45- 76  * Y1- .241 V1- .729 A1- .623 H1- .632 B1- .667
46- 77  * Y1- .254 V1- .712 A1- .607 H1- .636 B1- .670
47- 78  * Y1- .243 V1- .732 A1- .758 H1- .736 B1- .750
48- 79  * Y1- .266 V1- .703 A1- .778 H1- .613 C1- .694
49- 80  * Y1- .271 V1- .711 A1- .774 H1- .620 T1- .685
50- 81  * Y1- .291 V1- .733 A1- .765 H1- .733 T1- .770
51- 82  * Y1- .331 V1- .773 A1- .814 H1- .653 T1- .689
52- 83  * Y1- .361 V1- .798 H1- .696 T1- .615 C1- .605
53- 84  * Y1- .428 V1- .796 T1- .831 D1- .772 C1- .836
54- 85  * D1- .672 Y1- .743 C1- .676 D1- .795 T1- .891
55- 86  * C1- .723 D1- .824 C1- .834 T1- .833 C1- .845
56- 87  * T1- .819 C1- .841 C1- .830 D1- .853 C1- .876
57- 88  * T1- .797 C1- .847 C1- .849 D1- .832 C1- .894
58- 89  * T1- .854 C1- .828 C1- .874 L1- .837 T1- .710
59- 90  * T1- .844 C1- .886 C1- .892 L1- .710 K1- .711
60- 91  * T1- .810 C1- .864 K1- .847 S1- .796 H1- .710
61- 92  * T1- .870 C1- .891 K1- .854 E1- .701 H1- .707
62- 93  * T1- .833 C1- .865 K1- .870 E1- .891 H1- .705
63- 94  * T1- .862 C1- .845 K1- .877 E1- .887 C1- .708
64- 95  * T1- .886 C1- .856 K1- .870 E1- .837 C1- .709
65- 96  * T1- .806 C1- .868 K1- .873 E1- .810 F1- .714
66- 97  * T1- .814 K1- .876 C1- .873 E1- .883 C1- .710
67- 98  * T1- .859 K1- .866 C1- .877 E1- .887 A1- .705
68- 99  * T1- .811 K1- .856 C1- .864 E1- .872 H1- .695
69-100  * T1- .854 C1- .838 K1- .871 F1- .872 L1- .694
70-101  * T1- .866 C1- .869 K1- .814 E1- .833 H1- .841
71-102  * T1- .854 C1- .876 K1- .802 L1- .822 C1- .824
72-103  * T1- .873 C1- .872 K1- .883 L1- .887 C1- .816
73-104  * T1- .806 C1- .858 K1- .820 L1- .878 F1- .889
74-105  * T1- .822 C1- .862 K1- .831 L1- .816 C1- .807
75-106  * T1- .848 C1- .858 K1- .895 L1- .897 C1- .852
76-107  * T1- .874 C1- .872 D1- .899 L1- .811 K1- .802
77-108  * T1- .860 C1- .856 D1- .880 L1- .837 K1- .810
78-109  * T1- .823 C1- .896 D1- .882 L1- .814 K1- .819
79-110  * T1- .843 C1- .891 D1- .835 L1- .822 C1- .823
80-111  * T1- .841 C1- .883 D1- .809 S1- .825 H1- .828
81-112  * T1- .870 C1- .896 D1- .813 S1- .829 E1- .835
82-113  * T1- .848 C1- .810 D1- .810 S1- .832 H1- .841
83-114  * D1- .819 T1- .826 C1- .823 S1- .840 H1- .847
84-115  * D1- .829 C1- .832 T1- .887 S1- .858 H1- .848
85-116  * T1- .863 C1- .858 F1- .872 L1- .850 T1- .879
86-117  * D1- .862 C1- .854 C1- .798 K1- .810 H1- .711
87-118  * T1- .729 C1- .833 C1- .736 R1- .847 K1- .748
88-119  * C1- .741 S1- .808 C1- .792 K1- .855 H1- .785
89-120  * S1-1.000 T1-1.000 C1-1.000 D1-1.000 T1-1.000
90-121  * S1-1.000 S1-1.000 C1-1.000 D1-1.000 F1-1.000
91-122  * S1-1.000 S1-1.000 C1-1.000 D1-1.000 T1-1.000
92-123  * S1-1.000 S1-1.000 C1-1.000 D1-1.000 T1-1.000
93-124  * S1-1.000 S1-1.000 C1-1.000 D1-1.000 F1-1.000
94-125  * S1-1.000 T1-1.000 C1-1.000 D1-1.000 T1-1.000
95-126  * S1-1.000 S1-1.000 F1-1.000 D1-1.000 T1-1.000
96-127  * S1-1.000 S1-1.000 C1-1.000 D1-1.000 T1-1.000
97-128  * S1-1.000 S1-1.000 F1-1.000 D1-1.000 T1-1.000
98-129  * S1-1.000 S1-1.000 C1-1.000 D1-1.000 T1-1.000
99-130  * S1-1.000 S1-1.000 C1-1.000 D1-1.000 T1-1.000
100-131 * S1-1.000 S1-1.000 C1-1.000 D1-1.000 F1-1.000
101-132 * S1-1.000 S1-1.000 C1-1.000 D1-1.000 F1-1.000
102-133 * S1-1.000 S1-1.000 C1-1.000 D1-1.000 F1-1.000
```

Vita

Roy Edward Bentkowski was born on 31 August 1955 in Detroit, Michigan. He graduated in 1973 from Grosse Pointe North High School in Michigan. He entered the University of Michigan, in Ann Arbor, where he received the degree of Bachelor of Electrical Engineering in April, 1977. Upon graduation he received a commission in the USAF through the ROTC program. He then entered the School of Engineering, Air Force Institute of Technology, in September 1977.

Permanent Address: 20711 Country Club
Harper Woods, MI 48225

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GE/EE/79-1	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* SEGMENTATION OF TOUCHING ENGLISH LETTERS		5. TYPE OF REPORT & PERIOD COVERED MS Thesis 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Roy Edward Bentkowski 2nd Lt.		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology(AFIT-EN) Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Foreign Technology Division/NIT Wright-Patterson AFB OH 45433		12. REPORT DATE March, 1979 13. NUMBER OF PAGES 133
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

Approved for public release; IAW AFR 190-17

Joseph P. Hipps, Major, USAF 16 MAY 1979
Director of Information

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Optical Character Recognition (OCR)
Pattern Recognition
Reading Machine
Segmentation Problem

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This paper examines the problem of building a machine to read uncontrolled type fonts set with essentially no space between letters (within words). The consequence of this type of data, which represents the usual format of printed text, is that the data vectors produced by the optical scanner contain multiple letters and/or fragments of letters that cannot be easily separated.

An algorithm based on a variant of running cross-correlation between prototype letters and successively "windowed" fragments of

over

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

STRACT (continued)

tence is employed. The algorithm computes the Euclidean
e between prototypes and the sentence fragment in a filtered
domain.
is shown that appropriate normalization and windowing
ues allow perfect recognition of touching letters within
This occurs even when no apriori knowledge of letter loca-
thin the word is available, provided that suitable proto-
an be established.
ltiple alphabet prototypes were then built and used to
widely differing type fonts. Techniques to set acceptance
lds were evaluated and the behavior of the resulting recog-
system tabulated. A number of false triggers did occur in
e and these were discussed. Recommendations for further
ments in the system are suggested.